Comparison of Adaptive Beamforming Algorithms for Automatic Speech Recognition

Diploma thesis

Universität Karlsruhe
Institut für Theoretische Informatik
Interactive Systems Laboratories

Author:
Ulrich Klee

SUPERVISORS:
Prof. Dr. Alex Waibel
Dr. John McDonough

May 30, 2006

Hiermit erkläre ich, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, 31.05.2006

Ulrich Klee

Abstract

Improvements in automatic speech recognizer systems have led towards a new generation of applications which can be controlled by a user's voice. The acceptance of this new input modality is highly depending on its naturalness for the user. Hands-free far field recordings are by far the least intrusive approaches and should therefore be preferred. To achieve an acceptable signal quality in these far field recordings, beamforming can be used to reduce noise, disturbances and reverberation. In this diploma thesis we compare three beamforming algorithms in the special context of automatic speech recognizer systems. We evaluate the classical delay and sum beamformer, an adaptive MVDR- and a new maximum likelihood HMM beamformer. We show how both adaptive algorithms can be formulated as a Kalman filter. Using a square root information filter implementation, we are able to add diagonal loading as well as process noise to the Kalman filter. We show, that the design of the HMM beamformer is especially well suited to improve recognition accuracy of ASR systems as it directly uses the feedback of the recognizer to adapt its beamforming weights.

Acknowledgements

I would like to thank Prof. Dr. Alex Waibel for the support of this diploma thesis and for giving me the opportunity to work and study at his Interactive Systems Laboratories at University Karlsruhe, Germany and at Carnegie Mellon University, Pittsburgh, PA, USA over the last three years. I would especially like to thank Dr. John McDonough who had the greatest influence on my development as Computer Scientist here at University Karlsruhe. The time he spend, helping me to solve problems or patiently explaining me new theories is countless. His constant support over the last three years was an invaluable gift. Last but not least I want to thank my parents. Their never-ending love and support during my life allowed me to grow up without serious worries and eventually made me the person I am today. This thesis is dedicated to them.

Contents

Chapter 1

Introduction

Improvements in automatic speech recognition have lately led to a growing number of applications which can be controlled by a users voice in addition to classical input devices like a keyboard or mouse. The acceptance of this new input modality is highly depending on its naturalness for the user. Close-talking microphones offer the best audio quality, but also force the user to focus on them, while using a device. In addition, they might further disturb the user and reduce his flexibility, if for example he has to wear a head-mounted microphone. Far-field microphones on the other hand offer the least intrusive technique to record a speakers voice but also increase the amount of additional noise and disturbances in the recordings. Beamforming offers one way to reduce this disadvantage in far-field recordings significantly.

Originally developed for radar and sonar, beamforming has been eventually adapted to the acoustic field. Given an array with multiple microphones, beamforming can be used to focus on an acoustic source while suppressing noise and disturbances from other directions. To measure the performance of beamformers, most papers and articles evaluated the gain in the *signal-to-noise ratio* (SNR) and used this criterion to optimize the beamformers. Unfortunately, improvements in the SNR do not necessarily transform into a better performance of an *automatic speech recognizer* (ASR).

All state of the art ASR systems do not use a plain signal for speech recognition but extract several features like Mel frequency cepstral coefficients, which are used instead. Seltzer [Seltzer03] was able to demonstrate that the SNR therefore is not an optimum criterion to optimize the output of a beamformer. Instead he suggested to incorporate the ASR system into the beamforming algorithms, using Mel features and the log likelihood of an ASR

hypothesis as a minimization criterion to optimize their performance. With this new approach called LIMABEAM, Seltzer could reduce WER compared to standard beamforming techniques significantly. His best approach used a *Fast Fourier Transformation* (FFT) filterbank and adapted the beamformer in the different subbands. Raub presented a similar approach [Raub04] with an FFT filterbank but working in the cepstral domain with comparable results.

In this diploma thesis we want to further investigate the performance of adaptive beamformers for ASR systems. We implemented two classical approaches and one new beamforming algorithm using a *cosine modulated filterbank* for all three beamformers. In contrast to the FFT filterbank used by Seltzer and Raub, this filterbank does not disturb the acoustic signal but allows a perfect reconstruction. We evaluate the three beamformers, namely a standard non-adaptive *delay and sum* (DS) beamformer, an adaptive *minimum variance distortionless response* (MVDR) and a new adaptive Maximum Likelihood *Hidden Markov Model* (HMM) *beamformer*, with a state of the art speech recognition system.

1.1 Outline

Chapter 2 gives an overview of classical beamforming techniques and introduces the DS- and MVDR beamformer. We show, how to formulate the adaptive MVDR beamformer as a *recursive least squares* (RLS) algorithm and introduce a technique called *diagonal loading* to increase the stability of beamformers.

As the RLS algorithm offers no easy way to implement diagonal loading, we introduce the concepts of Kalman filtering in Chapter 3. We present a Kalman filter based on a square root algorithm and show how it can be used to implement an adaptive MVDR beamformer with diagonal loading. Chapter 4 focuses on a new approach specially designed for automatic speech recognition - the HMM beamformer. One important aspect of this diploma thesis is our implementation and first evaluation of this new beamformer.

Before deriving the HMM beamformer, we briefly present the concepts of modern ASR systems and the features used to recognize speech. These concepts will play an important role in a gradient derivation, which we can use to optimize the HMM beamformer performance.

The experimental setup and the results for the different algorithms are

presented in Chapter 5 as is a description of the ASR system used in our experiments. A final conclusion and motivation for future work is the topic of Chapter 6. Appendix A describes filterbanks and their advantages in beamforming applications. Furthermore, it introduces the cosine modulated filterbank; a perfect reconstruction filterbank, that was used in our beamforming experiments.

Chapter 2

Classical Beamforming Algorithms

The main goal of beamforming is to focus on one audio target while trying to suppress noise and disturbances from other directions. This chapter gives an introduction to beamforming and describes two of the three algorithms we used in our evaluation, the DS beamformer and the MVDR beamformer. Most of the examples and derivations in this chapter are taken from Van Trees [VanTrees02]. Both beamformers are designed to improve the SNR of a recording, while the HMM beamformer in Chapter 4 is specially designed to improve the output of an ASR system.

The DS beamformer is one of the oldest classical approaches, but still widely used for beamforming applications. With a complexity of $O(N)$ it is fast enough for realtime applications. The MVDR beamformer on the other hand can be formulated as an adaptive algorithm. It is able to deal with changing scenarios at the cost of a higher complexity.

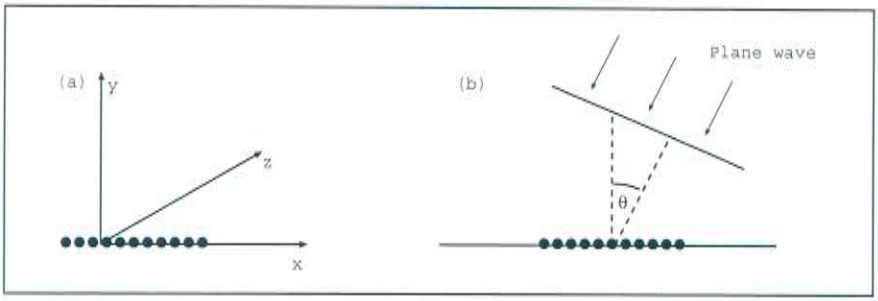
In all examples given throughout this chapter we will use a linear microphone array with equidistant spacing as used in our experimental setup.

2.1 Delay and Sum Beamformer

The DS beamformer is one of the oldest algorithms in beamforming, yet it is still the reference to other approaches as it is computationally simple as well as stable and reliable under most acoustic conditions.

Throughout this diploma thesis, we assume the microphone array to be

Figure 2.1: Linear array and plane wave



linear, located on the x-axis of a 3 dimensional room, as shown in Figure 2.1(a). Reducing the observation space to two dimensions, the incoming signal is assumed to be a plane wave from *direction* θ as shown in Figure 2.1(b).

Given a linear array with N microphones, the position of each microphone is:

$$p_n = \left(n - \frac{N-1}{2}\right) \cdot d$$

where d is the *inter-sensor distance*. The input y_n at each microphone is a filtered and time shifted version of the original speaker output:

$$\mathbf{y_n}(t) = \mathbf{x}(t) * \mathbf{h_n}(t) \tag{2.1}$$

where $*$ is the convolution formula, $x(t)$ is the original signal and $h_n(t)$ is the room impulse response

$$\mathbf{h_n}(t) = \frac{1}{N}\,\delta(t + \tau_n)$$

with τ_n as the *time delay* for the n-th microphone. If we know the position of the speaker and microphones, the *time delays of arrival* (TDOAs) τ_n can be calculated by

$$\tau_n = \frac{\|\mathbf{p_n} - \mathbf{s}\|}{v} = \frac{\sqrt{(p_{x_n} - s_x)^2 + (p_{y_n} - s_y)^2 + (p_{z_n} - s_z)^2}}{v} \tag{2.2}$$

where $\mathbf{p_n}$ is the position vector of the n-th microphone, $\mathbf{s}$ the position of the acoustic source and v a scalar, indicating the speed of sound through air.

For a *narrowband* signal with center frequency ω_c the time delay corresponds to a phase shift, which can be applied by the multiplication of $e^{j\omega_c \tau_n}$ in the frequency domain. Transforming (2.1) in the frequency domain, the convolution becomes a simple multiplication:

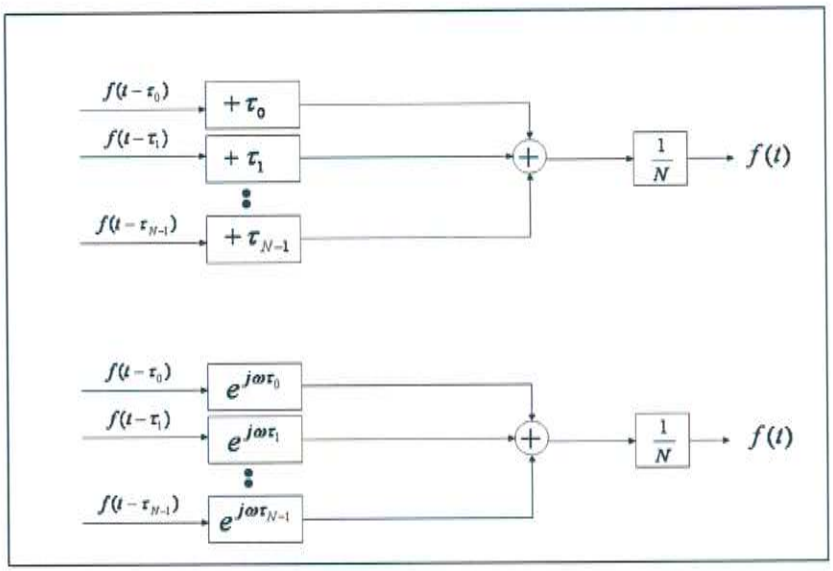
$$\mathbf{Y}_n(\omega) = \mathbf{X}(\omega)\ H_n(\omega),$$

with frequency response

$$H_n(\omega_c) = \frac{1}{N} \cdot e^{j\omega_c \tau_n}.$$

Figure 2.2 shows the two approaches working either in the time or frequency domain.

Figure 2.2: DS Beamformer [VanTrees02, Chap. 2]

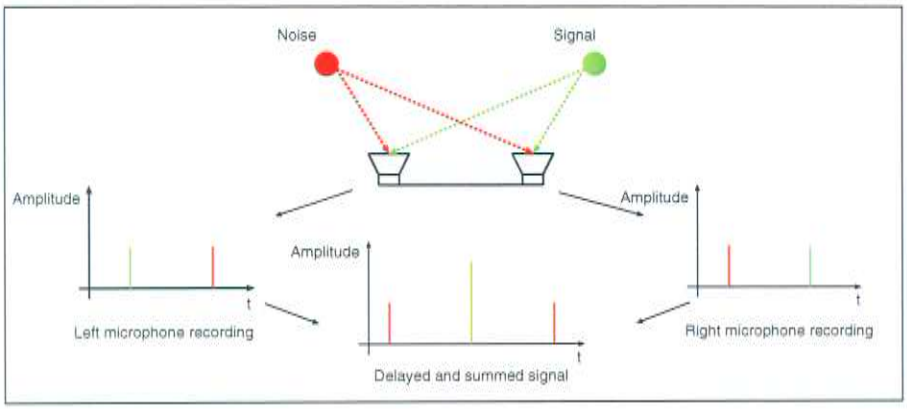


The DS beamformer improves the SNR by compensating this time delay and summing up all microphone channels. If the time delay is calculated correctly, the different signals of the speaker will overlap while noise from other directions won't.

Figure 2.3 demonstrates this idea: An acoustic signal and a noise source are located at different positions in front of two microphones. The signal is closer to the right one, hence, this microphone will record the signal earlier

than the left one. Similar, the noise source will be recorded earlier by the left microphone. The DS beamformer compensates the time delay between the two microphones for the signal, and sums up both channels. As the recording of the signal is now overlapping, the amplitude of the signal is almost doubled while the amplitude of the noise source stays the same. This results in a SNR almost twice as high as in the original signal.

Figure 2.3: Delayed and summed signal



The interactions of the array with any plane wave arriving at the microphones can be summarized by an *array manifold vector* $\mathbf{v}$:

$$\mathbf{v}^H(k_z) = \left[e^{j\mathbf{p}^{\mathbf{T}}{}_0\mathbf{k}_z} \mid e^{j\mathbf{p}_1^T\mathbf{k}_z} \mid \ldots \mid e^{j\mathbf{p}_n^T\mathbf{k}_z} \right]$$

with *wavenumber* $k_z = -\frac{2\pi}{\lambda}\cos\theta$; the direction of an arriving plane wave with wave length λ. Given the array manifold vector, the output F of a DS beamformer is the inner product of $\mathbf{X}$ and $\mathbf{v}$:

$$F(\omega) = \mathbf{X}^H(\omega)\mathbf{v}(k_z)$$

2.1.1 Computational Complexity

The computational complexity for the DS beamformer is very low. Beside the calculations inside the analysis and synthesis filterbank, the beamformer just requires the calculation of the array manifold vector $\mathbf{v}(k_z)$ and an inner product between $\mathbf{v}(k_z)$ and the input signal $\mathbf{X}(\omega)$. The complexity is growing linear in $O(N)$, where N is the number of microphones in the array. It is

therefore perfectly usable for real time applications even with a high number of microphones.

2.2 Beampatterns

The effect of a beamformer can be visualized by a so called beampattern [VanTrees02]. A beampattern shows the response of a beamformer *steered* into a certain direction where steering can be applied by the phase shift. The beampattern is given by

$$B(\theta) = e^{-j\left(\frac{N-1}{2}\right)\frac{2\pi d}{\lambda}\cos\theta} \sum_{n=0}^{N-1} w_n^* e^{jn\frac{2\pi d}{\lambda}\cos\theta}, \tag{2.3}$$

where w_n describes the beam weight of the n-th microphone. With $u = \cos\theta$ equation (2.3) becomes

$$B(u) = e^{-j\left(\frac{N-1}{2}\right)\frac{2\pi d}{\lambda}u} \sum_{n=0}^{N-1} w_n^* e^{jn\frac{2\pi d}{\lambda}u}, \tag{2.4}$$

and with $\psi = \frac{2\pi d}{\lambda}u$, (2.4) can be further reduced to

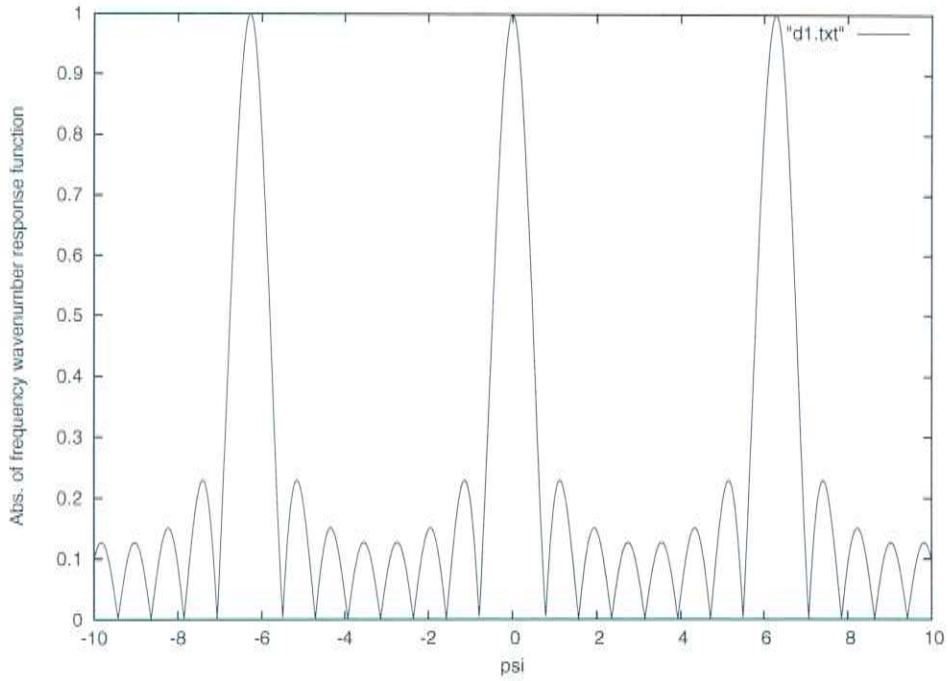
$$B(\psi) = e^{-j\left(\frac{N-1}{2}\right)\psi} \sum_{n=0}^{N-1} w_n^* e^{jn\psi}. \tag{2.5}$$

Let us show some relevant properties of a beampattern on the example of the DS beamformer. As stated in Section 2.1, the DS beamformer uses uniform weighting, hence we set the beamforming weights to $w_n^* = \frac{1}{N}$. This reduces (2.5) to

$$\begin{aligned} B_{DS}(\psi) &= e^{-j\left(\frac{N-1}{2}\right)\psi} \sum_{n=0}^{N-1} \frac{1}{N} e^{jn\psi} \\ &= \frac{1}{N} e^{-j\left(\frac{N-1}{2}\right)\psi} \sum_{n=0}^{N-1} e^{jn\psi}. \end{aligned}$$

Figure 2.4 shows the beampattern of a DS beamformer in ψ-space, steered perpendicularly to the array. At $\psi = 0$ a *main lobe* with high response is

Figure 2.4: Beampattern of a DS beamformer



visible. This main lobe should always be steered towards the acoustic source, in order to process the signal without distortion; this is equal to a frequency wavenumber response of one. Several *sidelobes* are located next to the main lobe. They have a smaller response than the main lobe but still the response is greater than zero, thus noise from these directions won't be completely suppressed. Beside those sidelobes you can see two other big lobes in the picture at approximately $\psi = -6.3$ and $\psi = 6.3$. They are called *grating lobes* and have the same magnitude as the main lobe. If a noise source matches the direction of a grating lobe, it won't be suppressed at all.

The *visible region* of a microphone array indicates the space where an acoustic source can be located. It is usually given by $-90^o \leq \theta \leq 90^o$ where $\theta = 0^o$ equals a look direction perpendicular to the array. In u-space this is similar to $-1 \leq u \leq 1$ and in ψ-space to $-\frac{2\pi d}{\lambda} \leq \psi \leq \frac{2\pi d}{\lambda}$.

The distance between main lobe and first grating lobe is directly affected by the interdistance d of the microphones. To avoid grating lobes inside the visible region, the interdistance between the microphones must be

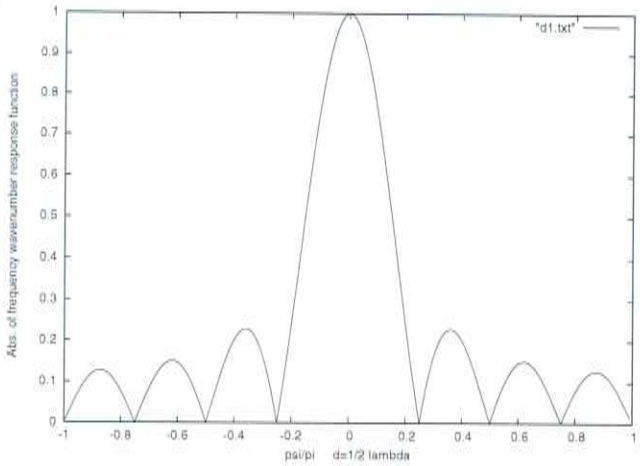
$$d \leq \frac{\lambda}{2}$$

where λ equals the wave length [Johnson93]. Figure 2.5 (a) to (c) shows the visible region of a DS beamformer with $d = \frac{\lambda}{2}$, $d = \lambda$ and $d = 2\lambda$. For $d = \lambda$ the first grating lobes appear at the outer rim of the visible region. As soon as the array is steered to another than the perpendicular direction, one of these grating lobes will move into the visible region, with a serious influence on the beamforming performance.

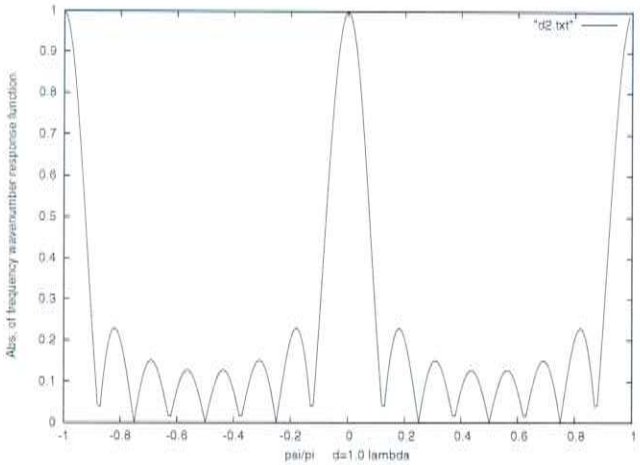
For uniformly weighted beamformers, the magnitude of the peak of the first side lobe is approximately -13.5 dB. This indicates the possible improvement for a uniformly weighted beamformer, like the standard DS beamformer, in the worst case, when the position of the noise source matches the first sidelobe. To further increase the SNR, it is necessary to influence the shape of the beampattern with respect to the speaker position and potential locations of noise sources. This is possible by adjusting the beam weights ω for each microphone non-uniformly. We will show how to get different beam weights in the following sections.

Figure 2.5: Influence of inter-microphone distance to location of grating lobes

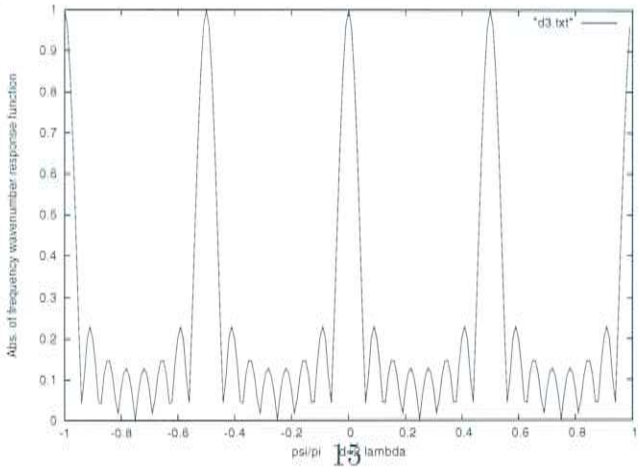
(a)



(b)



(c)



2.3 Generalized Sidelobe Canceler

A *generalized sidelobe canceler* (GSC) [Griffiths82] is a popular adaptive array design which is capable of adjusting the beam weights for each microphone in the array differently. The goal of the GSC is to process the signal of the active sound source (e.g. a speaker) without distortion (*distortionless constraint* [VanTrees02]), while trying to adjust the weights for each channel according to another optimization criterion, e.g. minimizing the total output power of the beamformer.

Given the N dimensional weight space (where N equals the number of microphones), the distortionless constraint is stated as

$$\mathbf{w}^H \, \mathbf{v}(k_s) = 1, \tag{2.6}$$

where $\mathbf{w}^H$ is the $1 \times N$ weight vector and $\mathbf{v}$ the $N \times 1$ array manifold vector with wavenumber k_s. Let us now divide the weight space into a *constraint* and an *orthogonal space*. The constraint space is represented by $\mathbf{v}$, the orthogonal space is spanned by $N-1$ linearly independent vectors of length N, which are orthogonal to $\mathbf{v}$. They can be summarized by the *blocking matrix* $\mathbf{B}$ of size $N \times (N-1)$. Thus,

$$\mathbf{B}^T \mathbf{v} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.7}$$

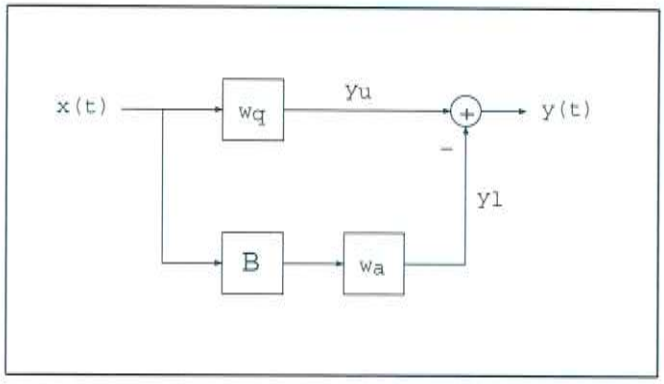
Let us assume that $\mathbf{w}_o$ is the optimum weight vector for the given acoustic condition. We can describe $\mathbf{w}_o$ by

$$\mathbf{w}_o = \mathbf{w}_q - \mathbf{B}\mathbf{w}_a,$$

where $\mathbf{w}_q$ is the projection of $\mathbf{w}_o$ on the constrained space and $\mathbf{w}_a$ the projection of $\mathbf{w}_o$ on the orthogonal space. We call $\mathbf{w}_q$ the *quiescent* weight vector which is determined by the distortionless constraint and $\mathbf{w}_a$ the *active* weight vector which has to be determined by some sort of optimization criterion. As $\mathbf{w}_a$ only affects the orthogonal space, we can manipulate $\mathbf{w}_a$ without affecting $\mathbf{w}_q$. We will show how to use different optimization criteria to calculate w_a with respect to the distortionless constraint in Section 2.4 and Section 4.2.

The total output of a GSC is the difference between an upper branch $y_u(\omega)$, which calculates the distortionless constrained output and a lower

Figure 2.6: Configuration of a generalized sidelobe canceler [VanTrees02]



branch $y_l(\omega)$ which works in the orthogonal space and usually tries to adapt to a noise source:

$$y_u(\omega) = \mathbf{x}^H(\omega)\mathbf{w}_q \tag{2.8}$$
$$y_l(\omega) = \mathbf{x}^H(\omega)\mathbf{B}\mathbf{w}_a \tag{2.9}$$
$$y(\omega) = y_u(\omega) - y_l(\omega). \tag{2.10}$$

Figure 2.6 shows the general configuration of a GSC.

It is possible to apply additional constraints like nulling out the signals in special directions. For M different constraints, (2.6) becomes

$$\mathbf{w}^H\ \mathbf{C} = \mathbf{g}$$

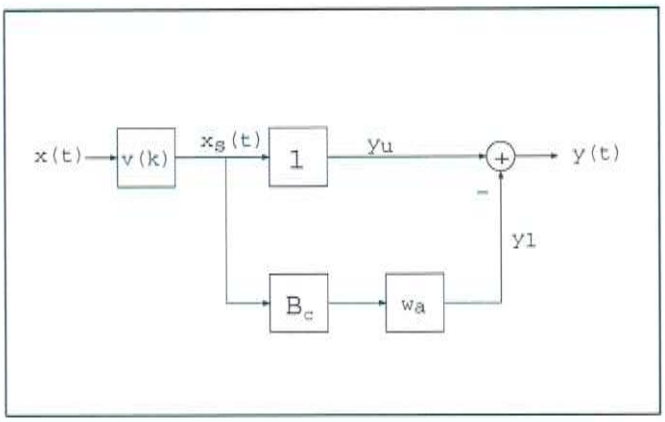
where $\mathbf{C}$ is an $N \times M$ *constrained matrix* and g an $1 \times M$ *constrained vector*. Similarly, (2.7) can be written as

$$\mathbf{B}^T \cdot \mathbf{C} = \mathbf{Z}$$

where $\mathbf{B}$ is the $N \times (N - M)$ blocking matrix and $\mathbf{Z}$ an $N \times (N - M)$ matrix with zeros. Obviously, with each additional constraint, the degree of freedom to manipulate w_a will be reduced by one. In the future discussion, we will focus on using only the distortionless constraint. Algorithms using other constraints are e.g. presented in Van Trees [VanTrees02].

Besides calculating $\mathbf{w}_a$, the computational complexity of a GSC depends on the elements of $\mathbf{C}$ and $\mathbf{B}$. In the worst case, we will have to recalculate $\mathbf{B}$ every time $\mathbf{C}$ changes. In our case, where $\mathbf{C}$ is zero except the first column

Figure 2.7: GSC with pre-steering



which is set to $\mathbf{v}(k_s)$, we would have to re-estimate $\mathbf{B}$ every time the speaker changes position as this changes the array manifold vector. We can avoid this recalculation by applying $\mathbf{v}$ to the input signal $\mathbf{x}$ before the beamforming. This will reduce $\mathbf{C}$ to a unity vector and allows us to use the constant blocking matrix

$$\mathbf{B}_c^T = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -1 \end{bmatrix}. \tag{2.11}$$

Equations (2.8)-(2.10) then become

$$\mathbf{x_s}(\omega) = \mathbf{x}^H(\omega)\,\mathbf{v}(k_s) \tag{2.12}$$

$$y_u(\omega) = \mathbf{x}_s^H(\omega) \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{2.13}$$

$$y_l(\omega) = \mathbf{x}_s^H(\omega)\,\mathbf{B}_c\mathbf{w}_a \tag{2.14}$$

$$y(\omega) = y_u(\omega) - y_l(\omega). \tag{2.15}$$

Figure 2.7 shows the GSC configuration with pre-steering. Notice that for all GSC configurations the output of a GSC is identical to a DS beamformer if $\mathbf{w}_a^H = [0 \cdots 0]$.

2.4 Minimum Variance Distortionless Response Beamformer

In this section we describe the non-adaptive MVDR beamformer. It is based on the assumption that the signal is nonrandom but unknown. The derivation of the adaptive MVDR approach follows in Section 2.5. All derivations are primarily based on Van Trees [VanTrees02].

As beamforming is usually done in the subband domain, we assume to have a *subband domain snapshot* $\mathbf{X}(\omega)$ of the input signal. The snapshot consists of frequency bins of the signal plus noise for each microphone channel

$$\mathbf{X}(\omega) = \mathbf{X}_s(\omega) + \mathbf{N}(\omega).$$

We get these snapshots of the original signal from a filter bank (in our case a cosined modulated one). Appendix A presents the cosine modulated filterbank and shows how to generate snapshots.

The signal vector can be written as

$$\mathbf{X}_s(\omega) = \mathbf{F}(\omega)\mathbf{v}(\omega : k_s),$$

where $\mathbf{F}(\omega)$ is the frequency-domain snapshot of the source signal and $\mathbf{v}(\omega : k_s)$ is the array manifold vector for a plane wave with wavenumber k_s.

The noise snapshot $\mathbf{N}(\omega)$ is a zero-mean random vector with *spatial spectral covariance matrix* $\mathbf{S}_n(\omega)$, defined as the sum between colored and white noise.

$$\mathbf{S}_n(\omega) = \mathbf{S}_c(\omega) + \sigma_\omega^2 \mathbf{I}. \tag{2.16}$$

Recall from Section 2.3 that we want to optimize the beamforming weights $\mathbf{w}_a$ in the orthogonal subspace with respect to the distortionless constraint

$$\mathbf{w}^H(\omega)\mathbf{v}(\omega : k_s) = 1. \tag{2.17}$$

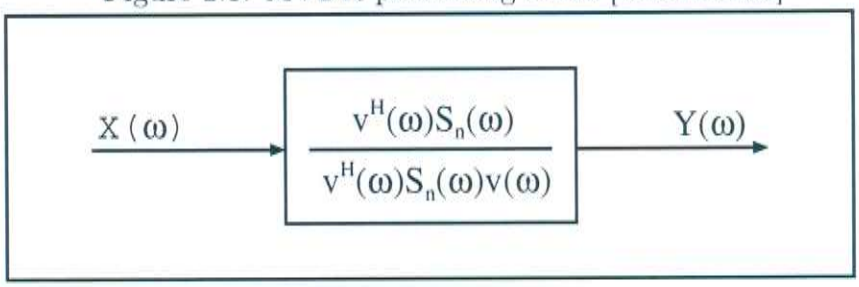
In the absence of noise, the distortionless constraint implies for the array output $\mathbf{Y}(\omega)$:

$$\mathbf{Y}(\omega) = \mathbf{F}(\omega)$$

for any $\mathbf{F}(\omega)$. Under this constraint, the optimization criterion for the lower branch of a GSC for the MVDR beamformer is to minimize the variance of $\mathbf{Y}(\omega)$ in the presence of noise. With

$$\mathbf{Y}(\omega) = \mathbf{F}(\omega) + \mathbf{Y}_n(\omega)$$

Figure 2.8: MVDR processing chain [VanTrees02]



this is equal to minimizing $E\left\{|\mathbf{Y}_n(\omega)|^2\right\}$.

The mean square of the output noise is

$$E\left\{|\mathbf{Y}_n(\omega)|^2\right\} = \mathbf{w}^H(\omega)\mathbf{S}_n(\omega)w(\omega). \tag{2.18}$$

We can solve for the optimal weights by applying the method of Lagrange multipliers:

$$\begin{aligned} F &= \mathbf{w}^H(\omega)\mathbf{S}_n(\omega)\mathbf{w}(\omega) + \lambda(\omega)\left[\mathbf{w}^H(\omega)\mathbf{v}(\omega : k_s) - 1\right] + \\ &\quad +\lambda^*(\omega)\left[\mathbf{v}^H(\omega : k_s)\mathbf{w}(\omega) - 1\right]. \end{aligned}$$

Taking the complex gradient with respect to $\mathbf{w}^H(\omega)$ and solving with respect to (2.17), it can be shown after some algebra, that

$$\mathbf{w}^H_{mvdr} = \frac{\mathbf{v}^H\mathbf{S}_n^{-1}}{\mathbf{v}^H\mathbf{S}_n^{-1}\mathbf{v}} \tag{2.19}$$

where we suppressed ω and k_s for convenience. Figure 2.8 shows the MVDR processing chain.

2.5 Adaptive MVDR Beamformer

The non-adaptive MVDR beamformer assumes that the spatial spectral covariance matrix $\mathbf{S}_n$ is known. In actual applications, this matrix has to be estimated from the incoming acoustic signal. Thus, the algorithm has to be changed to be able to adapt while receiving new data. Given the noise snapshots $\mathbf{N}$, we can estimate the spacial spectral matrix $\mathbf{S}_n$ with

$$\widehat{\mathbf{S}}_n = \frac{1}{T}\sum_{t=1}^{T}\mathbf{N}(t)\mathbf{N}^H(t), \tag{2.20}$$

hence equation (2.19) becomes

$$\hat{\mathbf{w}}_{mvdr}^{H} = \frac{\mathbf{v}^{H}\widehat{\mathbf{S}}_{n}^{-1}}{\mathbf{v}^{H}\widehat{\mathbf{S}}_{n}^{-1}\mathbf{v}}. \tag{2.21}$$

For the MVDR beamformer, we must only adapt the beamformers weights, when no desired speech is present. If we update the beamformer when speech is present, there is a risk that the algorithm will assume the signal to be noise and try to cancel it out. A *speech activity detector* can be used to indicate whether a speech signal is present or not.

If we estimate $\hat{\mathbf{S}}_n$ according to (2.21), all noise snapshots from the distant past up to now have the same influence to spatial spectral matrix. It would be more convenient if we could weight the new update. We therefore present two recursive algorithms which allow this sort of weighting.

First, we introduce the *recursive least squares* (RLS) algorithm. It uses a *forgetting factor* to weight the influence of past estimations to the current one. After that, we show how measurement mistakes either in speaker position estimations or in the microphone configuration can lead to instability of the beamformer and present a solution called *diagonal loading*. To apply this diagonal loading we will find it useful to transform the RLS equations into a Kalman filter, presented in Chapter 3.

2.6 Recursive Least Squares Formulation

The RLS algorithm offers the possibility to update a state, using just the old state estimate from the last time step and the input. The influence of past values and the current update is weighted by a fixed factor.

Let us rewrite (2.20):

$$\mathbf{\Phi}(T) = \sum_{t=1}^{T} \mu^{T-t}\mathbf{N}(t)\mathbf{N}^{H}(t), \tag{2.22}$$

where the matrix $\mathbf{\Phi}(t)$ represents an exponentially weighted sample spectral matrix [VanTrees02] and $\mu\epsilon(0,1)$ is a forgetting factor, which is usually set close to one. The forgetting factor indicates the influence of past observations to the current state.

For a recursive update of $\mathbf{\Phi}(t)$ we can rewrite (2.22):

$$\mathbf{\Phi}(t) = \mu\mathbf{\Phi}(t-1) + \mathbf{N}(t)\mathbf{N}^{H}(t). \tag{2.23}$$

Formula (2.21) requires the inversion of $\hat{\mathbf{S}}_n$ (or $\mathbf{\Phi}(t)$ in case of an recursive update), which can be calculated according to the matrix inversion lemma.

Matrix Inversion Lemma: *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two positive definite* $N \times N$ *matrices related by*

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}^T \mathbf{D}^{-1} \mathbf{C}$$

where $\mathbf{D}$ *is a positive-definite* $L \times L$ *matrix and* C *is an* $N \times L$ *matrix. Then the inverse of* $\mathbf{A}$ *can be expressed as*

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}^T \left(\mathbf{D} + \mathbf{C}\mathbf{B}\mathbf{C}^T\right)^{-1} \mathbf{C}\mathbf{B}$$

Using the matrix inversion lemma with relations

$$\begin{aligned} \mathbf{\Phi}(t) &= \mathbf{A} \\ \mu\mathbf{\Phi}(t-1) &= \mathbf{B}^{-1} \\ \mathbf{N}(t) &= \mathbf{C}^T \\ 1 &= \mathbf{D} \end{aligned}$$

we can invert (2.23):

$$\mathbf{\Phi}^{-1}(t) = \mu^{-1}\mathbf{\Phi}^{-1}(t-1) - \frac{\mu^{-2}\mathbf{\Phi}^{-1}(t-1)\mathbf{N}(t)\mathbf{N}^H(t)\mathbf{\Phi}^{-1}(t-1)}{1 + \mu^{-1}\mathbf{N}^H(t)\mathbf{\Phi}^{-1}(t-1)\mathbf{N}(t)}. \tag{2.24}$$

Defining

$$\mathbf{P}(t) = \mathbf{\Phi}^{-1}(t), \tag{2.25}$$

and

$$\mathbf{g}(t) = \frac{\mu^{-1}\mathbf{P}(t-1)\mathbf{N}(t)}{1 + \mu^{-1}\mathbf{N}^H(t)\mathbf{P}(t-1)\mathbf{N}(t)} \tag{2.26}$$

we can rewrite (2.24) to

$$\mathbf{P}(t) = \mu^{-1}\mathbf{P}(t-1) - \mu^{-1}\mathbf{g}(t)\mathbf{N}^H(t)\mathbf{P}(t-1), \tag{2.27}$$

which is the well known Ricatti equation[1]. The Kalman filter we present in Section 3.1 is also based on the Ricatti equation, although in a form that must be modified to account for the process noise.

[1] Named after Count Jacopo Francisco Riccati

2.7 Diagonal Loading

Up to this point, we had assumed that we exactly knew the position of the speaker and the microphone array. We have also assumed that the inter-distance between two microphones is exactly the same. Under realistic conditions it is most likely that there are inaccuracies in the position of the sensors and the estimated position of the speaker. These inaccuracies lead to a reduced performance of any beamformer. In the worst case, when the signal and noise source location is completely wrong, we might even cancel out the signal of the speaker and enhance the disturbances from the noise source. As shown in [VanTrees02, p. 505 ff], the sensitivity of beamformers to these inaccuracies increases as $||\mathbf{w}||^2$ increases. VanTrees demonstrates, that it is useful to apply a quadratic constraint

$$||\mathbf{w}||^2 \leq T_0, \tag{2.28}$$

with T_0 as a design parameter to limit the magnitude of $\mathbf{w}_a$.

As stated in Section 2.4, the MVDR beamformer tries to minimize the variance of the array output in the presence of noise and with respect to the distortionless constraint (2.17). Thus, VanTrees adds to the Lagrange multiplier function a term corresponding to the quadratic constraint:

$$\mathbf{w}^H\mathbf{w} - T_0 = 0,$$

which leads to

$$\begin{aligned} F &= \mathbf{w}^H\mathbf{S}_n\mathbf{w} + \lambda_1\left[\mathbf{w}^H\mathbf{w} - T_0\right] + \\ &\quad +\lambda_2\left[\mathbf{w}^H\mathbf{v} - 1\right] + \lambda_2^*\left[\mathbf{v}^H\mathbf{w} - 1\right]. \end{aligned} \tag{2.29}$$

Differentiating (2.29) with respect to λ_2^* and equating to zero gives

$$\mathbf{w}_{mvdr}^H = \mathbf{w}^H = \frac{\mathbf{v}^H\left[\hat{\mathbf{S}}_n + \lambda_1\mathbf{I}\right]^{-1}}{\mathbf{v}^H\left[\hat{\mathbf{S}}_n + \lambda_1\mathbf{I}\right]^{-1}\mathbf{v}}. \tag{2.30}$$

Hence, applying the quadratic constraint is equal to adding a diagonal matrix to the spatial spectral matrix $\mathbf{S}_n$. This is called *diagonal loading*. The value of λ_1 depends on the choice of T_0. It can either be set to a fixed value or be

adapted with the incoming signal. In general, the active weights of a GSC beamformer will get smaller with increasing values for λ_1.

To prove this statement, we solve (2.29) with respect to $\mathbf{w}$ and set the result to zero:

$$\mathbf{w}^H \mathbf{S}_n + \lambda_1 \mathbf{w}^H + \lambda_2 \mathbf{v}^H = 0,$$

or

$$\mathbf{w}^H = -\lambda_2 \mathbf{v}^H \left[\lambda_1 \mathbf{I} + \mathbf{S}_n\right]^{-1}.$$

Solving for λ_2 gives

$$\mathbf{w}^H = \frac{\mathbf{v}^H \left[\mathbf{S}_n + \lambda_1 \mathbf{I}\right]^{-1}}{\mathbf{v}^H \left[\mathbf{S}_n + \lambda_1 \mathbf{I}\right]^{-1} \mathbf{v}}. \tag{2.31}$$

With

$$\mathbf{w}_{gsc} = \mathbf{w}_q - \mathbf{B}\mathbf{w}_a$$

and

$$\mathbf{B}^H \mathbf{B} = \mathbf{I},$$

equation (2.31) can be used to derive

$$\mathbf{w}_a = \left[\mathbf{B}^H \mathbf{S}_n \mathbf{B} + \lambda_1 \mathbf{I}\right]^{-1} \mathbf{B}^H \mathbf{S}_n \mathbf{w}_q. \tag{2.32}$$

To show that the norm of $\mathbf{w}_a$ decreases as λ_1 increases, let us rewrite (2.32). By setting

$$\begin{aligned} \mathbf{S}_z &= \mathbf{B}^H \mathbf{S}_n \mathbf{B} \\ \mathbf{p}_z &= \mathbf{B}^H \mathbf{S}_n \mathbf{w}_q, \end{aligned}$$

we get

$$\mathbf{w}_a = \left[\mathbf{S}_z + \lambda_1 \mathbf{I}\right]^{-1} \mathbf{p}_z,$$

The squared norm of w_a is then given by

$$\mathbf{w}_a^H \mathbf{w}_a = \mathbf{p}_z^H \left(\mathbf{S}_z + \lambda_1 \mathbf{I}\right)^{-2} \mathbf{p}_z. \tag{2.33}$$

Taking the partial derivative of (2.33) with respect to λ_1 gives

$$\frac{\partial \mathbf{w}_a^H \mathbf{w}_a}{\partial \lambda_1} = -2\mathbf{p}_z^H \left(\mathbf{S}_z + \lambda_1 \mathbf{I}\right)^{-3} \mathbf{p}_z. \tag{2.34}$$

For $\lambda_1 > 0$, the diagonally loaded matrix $\left[\mathbf{S}_z + \lambda_1 \mathbf{I}\right]$ is positive definite, hence $\mathbf{p}_z^H \left(\mathbf{S}_z + \lambda_1 \mathbf{I}\right)^{-3} \mathbf{p}_z$ is positive, therefore (2.34) is always negative. Thus, increasing λ_1 decreases the norm of $\mathbf{w}_a$ [VanTrees02].

Without diagonal loading it is very likely that the beamforming weights will grow above all reasonable values and the beamformer will become unstable. The RLS algorithm offers no easy way to directly add diagonal loading, though several approximations for this problem exist.

In contrast to RLS algorithms, Kalman filters are based on a state-space equation which includes process noise in its calculation. We will show how we can easily apply diagonal loading to $\mathbf{S}_n$ by adding a diagonal matrix to the inverse of $\mathbf{S}(t+1|t)$, when formulating a Kalman filter as innovation filter. The next chapter gives an introduction to Kalman filters and shows how to implement the MVDR beamformer with this technique.

Chapter 3

Kalman Filter

This chapter summarizes the basic ideas of Kalman filters, following the description in [Haykin96]. It demonstrates how Kalman filters based on the Ricatti equation can become unstable and presents a solution to this problem with a square root implementation. After introducing the theoretical backround we show how to implement an adaptive MVDR beamformer as a Kalman filter.

3.1 Kalman Filter Based on the Ricatti Equation

The basic idea behind Kalman filters is the formulation of a state-space model consisting of a *process* and an *observation* equation

$$\begin{aligned} \mathbf{X}(t+1) &= \mathbf{A}(t)\mathbf{X}(t) + \mathbf{W}(t) && (3.1) \\ \mathbf{Y}(t) &= \mathbf{C}(t)\mathbf{X}(t) + \mathbf{V}(t). && (3.2) \end{aligned}$$

The only observable value is the *observation* $\mathbf{Y}(t)$ while the *state* $\mathbf{X}(t)$ is not observable but has to be estimated. Inside the process equation, $\mathbf{X}(t+1)$ is calculated by multiplying $\mathbf{X}(t)$ with *transition matrix* $\mathbf{A}(t)$ and adding *process noise* $\mathbf{W}(t)$. The next observation can be predicted by multiplying state $\mathbf{X}(t)$ with *observation matrix* $\mathbf{C}(t)$ and adding *observation noise* $\mathbf{V}(t)$. Both observation and process noise are assumed to be zero mean, white random processes and statistically independent.

Based on these formulae the *predicted state estimate* $\hat{\mathbf{X}}(t+1|t)$ is defined in addition to the *filtered* state $\mathbf{X}(t)$. It can be calculated by

$$\hat{\mathbf{X}}(t+1|t) = \mathbf{A}(t)\hat{\mathbf{X}}(t|t)$$

Also define the *predicted observation vector* $\hat{\mathbf{Y}}(t+1|t)$, which can be calculated by

$$\hat{\mathbf{Y}}(t+1|t) = \mathbf{C}(t)\hat{\mathbf{X}}(t+1|t) \tag{3.3}$$

With (3.3) Haykin formulates the *innovation term* as the difference between the filtered and the predicted observation

$$\begin{aligned} \alpha(t) &= \mathbf{Y}(t) - \hat{\mathbf{Y}}(t|t-1) && (3.4)\\ &= \mathbf{Y}(t) - \mathbf{C}(t-1)\hat{\mathbf{X}}(t|t-1) && (3.5) \end{aligned}$$

The correlation matrix of the innovation term can be formulated as

$$\begin{aligned} \mathbf{F}(t) &= E\left\{\alpha(t)\alpha^T(t)\right\} \\ &= \mathbf{C}(t)\mathbf{K}(t|t-1)\mathbf{C}^T(t) + \mathbf{R}(t), \end{aligned}$$

where $\mathbf{K}(t|t-1)$ is the correlation matrix of the *predicted state error*

$$\begin{aligned} \epsilon(t|t-1) &= \mathbf{X}(t) - \hat{\mathbf{X}}(t|t-1) \\ \mathbf{K}(t|t-1) &= E\left\{\epsilon(t|t-1)\epsilon^H(t|t-1)\right\} && (3.6) \end{aligned}$$

and $\mathbf{R}(t)$ equals the correlation matrix of the observation noise vector $\mathbf{V}$. The correlation matrix $\mathbf{K}(t)$ of the *state error* $\epsilon(t)$ is given by

$$\begin{aligned} \epsilon(t) &= \mathbf{X}(t) - \hat{\mathbf{X}}(t|t) \\ \mathbf{K}(t) &= E\left\{\epsilon(t)\epsilon^H(t)\right\}. && (3.7) \end{aligned}$$

The *Kalman gain* is defined as

$$\begin{aligned} \mathbf{G}(t) &= E\left\{\hat{\mathbf{X}}(t+1|t)\alpha^T(t)\right\}\mathbf{F}^{-1}(t) \\ &= \mathbf{A}(t)\mathbf{K}(t|t-1)\mathbf{C}^T(t)\mathbf{F}^{-1}(t). && (3.8) \end{aligned}$$

It is a weighting factor for the next state update, depending on the confidence of the last state estimation. Before we can calculate $\mathbf{G}(t)$ we have to know the correlation matrix $\mathbf{K}(t|t-1)$. It can be estimated with the *Ricatti equation*

$$\begin{aligned} \mathbf{K}(t+1|t) &= \mathbf{A}(t)\mathbf{K}(t)\mathbf{A}^T(t+1|t) + \mathbf{Q}(t) && (3.9)\\ \mathbf{K}(t) &= \mathbf{K}(t|t-1) - \mathbf{A}(t)\mathbf{G}(t)\mathbf{C}(t)\mathbf{K}(t|t-1) \\ &= \left[\mathbf{I} - \mathbf{A}(t)\mathbf{G}(t)\mathbf{C}(t)\right]\mathbf{K}(t|t-1). && (3.10) \end{aligned}$$

Figure 3.1: Graph of a one step prediction Kalman filter [Haykin96]

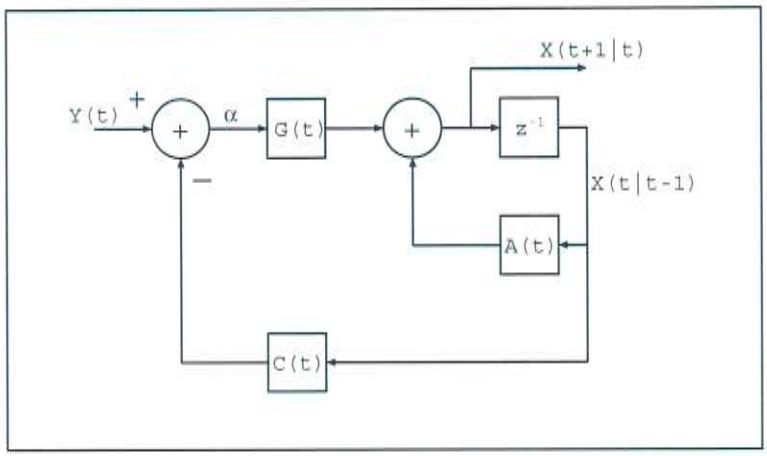


Figure 3.1 shows the signal flow graph of a one step prediction Kalman filter and Algorithm 1 summarizes the calculations for a Kalman filter based on the Ricatti equation.

3.2 Extended Kalman Filter

Kalman filters based on the Ricatti equation assume the process matrix $\mathbf{A}(t)$ and the observation matrix $\mathbf{C}(t)$ to be linear. Though this assumption holds true for the MVDR beamformer, we will show that it is not true for our third algorithm: The HMM beamformer, that we will derive in Chapter 4. Its measurement matrix is nonlinear and can therefore not be used in a standard Kalman filter.

With nonlinear measurement matrix, the state space formulae become

$$\begin{aligned} \mathbf{X}(t+1) &= \mathbf{A}(t)\mathbf{X}(t) + \mathbf{W}(t) \\ \mathbf{Y}(t) &= \mathbf{C}\left(t, \mathbf{X}(t)\right) + \mathbf{V}(t). \end{aligned}$$

The *extended Kalman filter* is an extension to the standard Kalman filter, which can deal with nonlinear observation and process matrices $\mathbf{C}\left(t, \mathbf{X}(t)\right)$ and $\mathbf{A}\left(t, \mathbf{X}(t)\right)$[1] [Haykin96]. It splits the state estimation formula into two steps, $\hat{\mathbf{X}}(t|t)$ and $\hat{\mathbf{X}}(t+1|t)$ and uses a partial derivative with respect to

[1] In our presentation we focus on non-linear observation matrices only.

Algorithm 1 Kalman filter based on Ricatti equation

Input process: $\mathbf{Y}(1), \mathbf{Y}(2), \ldots, \mathbf{Y}(t)$
Known parameters:

- Transition matrix $\mathbf{A}(t)$
- Observation matrix $\mathbf{C}(t)$
- Covariance matrix of process noise $\mathbf{Q}(t) = E\left\{\mathbf{W}(t)\mathbf{W}^T(t)\right\}$
- Covariance matrix of observation noise $\mathbf{R}(t) = E\left\{\mathbf{V}(t)\mathbf{V}^T(t)\right\}$

Computation $t = 1, 2, 3, \ldots$

$$\begin{aligned}
\mathbf{G}(t) &= \mathbf{A}(t)\mathbf{K}(t|t-1)\mathbf{C}^T(t)\left[\mathbf{C}(t)\mathbf{K}(t|t-1)\mathbf{C}^T(t) + \mathbf{R}(t)\right]^{-1} \\
\alpha(t) &= \mathbf{Y}(t) - \hat{\mathbf{Y}}(t|t-1) \\
\hat{\mathbf{X}}(t+1|t) &= \mathbf{A}(t)\hat{\mathbf{X}}(t|t-1) + \mathbf{G}(t)\alpha(t) \\
\mathbf{K}(t) &= \left[\mathbf{I} - \mathbf{A}(t)\mathbf{G}(t)\mathbf{C}(t)\right]\mathbf{K}(t|t-1) \\
\mathbf{K}(t+1|t) &= \mathbf{A}(t)\mathbf{K}(t)\mathbf{A}^T(t) + \mathbf{Q}(t)
\end{aligned}$$

$\hat{\mathbf{X}}(t|t-1)$ to linearize the non-linear measurement matrix

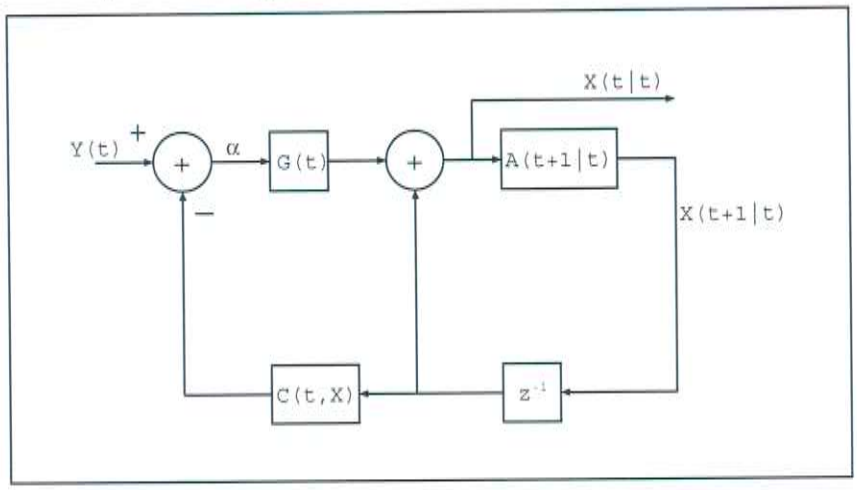
$$\mathbf{C}(t) = \left.\frac{\partial \mathbf{C}(t, \mathbf{X})}{\partial \mathbf{X}}\right|_{\mathbf{X}=\hat{\mathbf{X}}(t|t-1)}. \tag{3.11}$$

Figure 3.2 shows the general configuration and Algorithm 2 shows the calculations for an extended Kalman filter.

3.3 Information Filter

The Kalman filtering equation can be alternatively formulated to use the inverse of the correlation matrices $\mathbf{K}(t)$ and $\mathbf{K}(t+1|t)$ respectively. These inverse matrices are called *information matrices* and the corresponding Kalman filter is named *information filter* [Fraser67]. The information filter has the advantage, that it can calculate updates even with very little information about the initial state of the system.

Figure 3.2: Signal flow of an extended Kalman filter



In our case, a second advantage arises: RLS estimators and Kalman filters are strongly related as shown for example in [Haykin96]. Comparing the RLS equation

$$\mathbf{P}(t) = \mu^{-1}\mathbf{P}(t-1) - \mu^{-1}\mathbf{g}(t)\mathbf{N}^H(t)\mathbf{P}(t-1)$$

from Section 2.6 with the necessary variables from the Kalman filter, namely the correlation matrix of the state error $\mathbf{K}(t)$, the Kalman gain $\mathbf{G}(t)$, the innovation factor $\alpha(t)$ and the predicted state vector $\hat{\mathbf{X}}(t+1|t)$, Haykin shows the following correspondences:

$$\begin{aligned} \mathbf{K}(t-1) &= \mu^{-1}\mathbf{P}(t-1) \\ \mathbf{G}(t) &= \mu^{-1/2}\mathbf{g}(t) \\ \alpha(t) &= \mu^{-n/2}\xi^*(t) \\ \hat{\mathbf{X}}(t+1|t) &= \mu^{-(t+1)/2}\hat{\mathbf{w}}(t), \end{aligned} \tag{3.12}$$

where $\mu^{-n/2}\xi^*(t)$ equals the *a-priori estimation error* and $\mu^{-(t+1)/2}\hat{\mathbf{w}}(t)$ is the estimate of the weight vector. With (3.10), we can see from (3.12) that $\mathbf{K}(t)$ equals the inverse of the spatial spectral covariance matrix $\mathbf{S}_n$ of the MVDR beamformer.

As stated in Section 2.7, we want to apply diagonal loading to increase the stability of the beamformer. This can be done by adding a diagonal matrix to the spatial spectral matrix $\mathbf{S}_n$. Hence if we used a standard Kalman filter,

Algorithm 2 Extended Kalman filter

Input process: $\mathbf{Y}(1), \mathbf{Y}(2), \ldots, \mathbf{Y}(t)$
Known parameters:

- Process matrix $\mathbf{A}(t)$
- Non-linear observation matrix $\mathbf{C}\left(t, \mathbf{X}(t)\right)$
- Covariance matrix of process noise $\mathbf{Q}(t) = E\left\{\mathbf{W}(t)\mathbf{W}^T(t)\right\}$
- Covariance matrix of observation noise $\mathbf{R}(t) = E\left\{\mathbf{V}(t)\mathbf{V}^T(t)\right\}$

Computation $t = 1, 2, 3, \ldots$

$$\begin{aligned}
\mathbf{G}(t) &= \mathbf{K}(t|t-1)\mathbf{C}^T(t)\left[\mathbf{C}(t)\mathbf{K}(t|t-1)\mathbf{C}^T(t) + \mathbf{R}(t)\right]^{-1} \\
\alpha(t) &= \mathbf{Y}(t) - \mathbf{C}\left(t, \hat{\mathbf{X}}(t|t-1)\right) \\
\hat{\mathbf{X}}(t|t) &= \hat{\mathbf{X}}(t|t-1) + \mathbf{G}(t)\alpha(t) \\
\mathbf{K}(t) &= \left[\mathbf{I} - \mathbf{G}(t)\mathbf{C}(t)\right]\mathbf{K}(t|t-1) \\
\mathbf{K}(t+1|t) &= \mathbf{A}(t)\mathbf{K}(t)\mathbf{A}^T(t) + \mathbf{Q}(t) \\
\hat{\mathbf{X}}(t+1|t) &= \mathbf{A}(t)\hat{\mathbf{X}}(t|t)
\end{aligned}$$

$\mathbf{C}(t)$ is calculated from the non-linear measurement matrix $\mathbf{C}\left(t, \mathbf{X}(t)\right)$ according to (3.11).

we would have to invert $\mathbf{K}(t)$ every time we want to apply diagonal loading:

$$\begin{aligned}
\mathbf{K}(t) &= \left[\mu\mathbf{\Phi}(t)\right]^{-1} = \hat{\mathbf{S}}_n^{-1}(t) \\
\mathbf{K}_{DL}(t) &= \left[\left(\mathbf{K}^{-1}(t)\right)^{-1} + \sigma_D^2\mathbf{I}\right]^{-1}.
\end{aligned}$$

As the innovation filter uses $\mathbf{K}^{-1}(t)$, we can directly add the diagonal matrix:

$$\begin{aligned}
\mathbf{K}^{-1}(t) &= \hat{\mathbf{S}}_n(t) \\
\mathbf{K}_{DL}^{-1}(t) &= \mathbf{K}^{-1}(t) + \sigma_D^2\mathbf{I}.
\end{aligned}$$

Algorithm 3 shows the calculations for the innovation filter based on the Ricatti equation. The write-up follows the NAG C Library description of time series analysis.

Algorithm 3 Information filter

Input process: $\mathbf{Y}(1), \mathbf{Y}(2), \ldots, \mathbf{Y}(t)$
Known parameters:

- Transition matrix $\mathbf{A}(t)$
- Observation matrix $\mathbf{C}(t)$
- Covariance matrix of process noise $\mathbf{Q}(t) = E\left\{\mathbf{W}(t)\mathbf{W}^T(t)\right\}$
- Covariance matrix of observation noise $\mathbf{R}(t) = E\left\{\mathbf{V}(t)\mathbf{V}^T(t)\right\}$

Computation $t = 1, 2, \ldots$:

$$
\begin{aligned}
\mathbf{K}^{-1}(t+1|t) &= [\mathbf{I} - \mathbf{P}(t)]\,\mathbf{M}(t) \\
\mathbf{K}^{-1}(t+1) &= \mathbf{K}^{-1}(t+1|t) + \mathbf{C}^T(t+1)\mathbf{R}^{-1}(t+1)\mathbf{C}(t+1) \\
\text{where } \mathbf{M}(t) &= \left(\mathbf{A}^{-1}(t)\right)^T K^{-1}(t)\mathbf{A}^{-1}(t) \\
\text{and } \mathbf{P}(t) &= \mathbf{M}(t)\left[\mathbf{Q}^{-1}(t) + \mathbf{M}(t)\right] \\
\hat{\mathbf{a}}(t+1|t) &= [\mathbf{I} - \mathbf{P}(t)]\left(\mathbf{A}^{-1}(t)\right)^T \hat{\mathbf{a}}(t) \\
\hat{\mathbf{a}}(t+1) &= \hat{\mathbf{a}}(t+1|t) + \mathbf{C}^T(t+1)\mathbf{R}^{-1}(t+1)\mathbf{Y}(t+1) \\
\text{where } \hat{\mathbf{a}}(t+1|t) &= \mathbf{K}^{-1}(t+1|t)\hat{\mathbf{X}}(t+1|t) \\
\text{and } \hat{\mathbf{a}}(t+1) &= \mathbf{K}^{-1}(t+1)\hat{\mathbf{X}}(t+1)
\end{aligned}
$$

3.3.1 Square Root Implementation of Innovation Filters

The Ricatti equation suffers from numerical instability. Numerical problems arise as in (3.10) the non-negative definite matrix $\mathbf{K}(t)$ is calculated as the difference of two other non-negative definite matrices. After multiple iterations it is therefore possible that $\mathbf{K}(t)$ loses its Hermitian symmetry and becomes indefinite.

To avoid this numerical problem a *square root* implementation can be used. A good overview of this technique is given in [Kaminski71, et al.]. As shown in [Dyer69], it is possible to extend the square root filtering algorithm to include process noise and therefore make it applicable for Kalman filters. The square root implementation requires the construction of two matrices:

The *pre-matrix*

$$\Gamma = \begin{pmatrix} \mathbf{Q}^{-1/2}(t) & 0 & 0 \\ \mathbf{K}^{-1/2}(t)\mathbf{A}^{-1}(t) & \mathbf{K}^{-1/2}(t) & \mathbf{K}^{-1/2}(t)\hat{\mathbf{X}}(t) \\ 0 & \mathbf{R}^{-1/2}(t+1)\mathbf{C}(t+1) & \mathbf{R}^{-1/2}(t+1)\mathbf{Y}(t+1) \end{pmatrix},$$

and the *post-matrix*

$$\Omega = \begin{pmatrix} \mathbf{F}^{-1/2}(t+1) & * & * \\ 0 & \mathbf{K}^{-1/2}(t+1) & \xi(t+1) \\ 0 & 0 & * \end{pmatrix}.$$

where $\mathbf{K}^{-1/2}, \mathbf{Q}^{-1/2}$ and $\mathbf{R}^{-1/2}$ are the upper triangular parts of the Cholesky factorized matrices $\mathbf{K}^{-1}$, $\mathbf{Q}^{-1}$ and $\mathbf{R}^{-1}$. The values in the post-matrix are defined as

$$\mathbf{F}^{-1}(t+1) = \mathbf{Q}^{-1}(t) + \left(\mathbf{A}^{-1}(t)\right)^T \mathbf{K}^{-1}(t)\mathbf{A}^{-1}(t) \quad (3.13)$$

$$\xi(t+1) = \mathbf{K}^{-1/2}(t+1)\hat{\mathbf{X}}(t+1). \quad (3.14)$$

All $*$ entries are not relevant for our algorithm.

A transformation matrix $\mathbf{U}_2$ is used to transform Γ into Ω

$$\mathbf{U}_2\Gamma = \Omega.$$

$\mathbf{U}_2$ is an orthogonal transformation, which restores the pre-matrix to upper triangular form. In so doing, the algorithm calculates the post-matrix, which contains all values necessary to perform the update in the (extended) Kalman Filter. This transformation can be easily achieved by a Householder transformation [Steinhardt88]. All values for the next time step can be directly obtained from the relevant blocks of the second matrix. As $\mathbf{K}^{-1/2}(t+1)$ is upper triangular, the inversion of $\mathbf{K}^{-1/2}(t+1)$ in (3.14) to get $\hat{\mathbf{X}}(t+1)$ is trivial.

3.4 Square Root Information MVDR Beamformer

With the relations between RLS filters and Kalman filters stated in Section 3.3, it is straight forward to transfer the MVDR algorithm formulated as RLS filter into a square root information filter. The pre-matrix becomes

$$\begin{pmatrix} \mathbf{Q}^{-1/2}(t) & 0 & 0 \\ \hat{\mathbf{S}}_n^{1/2}(t) & \hat{\mathbf{S}}_n^{1/2}(t) & \hat{\mathbf{S}}_n^{1/2}(t)\mathbf{w}_a(t) \\ 0 & \mathbf{R}^{-1/2}(t+1)\mathbf{y}_l(t+1) & \mathbf{R}^{-1/2}(t+1)\mathbf{y}_u(t+1) \end{pmatrix}$$

and the post-matrix is given by

$$\begin{pmatrix} \mathbf{F}^{-1/2}(t+1) & * & * \\ 0 & \hat{\mathbf{S}}_n^{1/2}(t+1) & \xi(t+1) \\ 0 & 0 & * \end{pmatrix},$$

where $\mathbf{y}_l$ and $\mathbf{y}_u$ are given by the upper and lower branch of a GSC and calculated as shown in Section 2.3.

As stated in Section 2.7, diagonal loading is necessary to increase robustness of the beamformer against signal delay-of-arrival mismatch and array perturbations. We showed, that this is equivalent to adding a diagonal matrix to the spacial spectral matrix $\hat{\mathbf{S}}_n$. Again, we can use the Householder transformation to perform the necessary calculations: By constructing one pre-matrix with $\hat{\mathbf{S}}_n^{1/2}(t+1)$ and the diagonal matrix $\sigma_D^2\mathbf{I}$ and performing another Householder transformation, we sweep the diagonal matrix into $\hat{\mathbf{S}}_n^{1/2}(t+1)$ and restore its upper triangular shape:

$$\underset{pre-matrix}{U_2 \begin{bmatrix} \hat{\mathbf{S}}_n^{1/2}(t+1) \\ \hline \sigma_D^2(t)\mathbf{I} \end{bmatrix}} = \underset{post-matrix}{\begin{bmatrix} \hat{\mathbf{S}}_{DL}^{1/2}(t+1) \\ \hline 0 \end{bmatrix}} \tag{3.15}$$

The complete computations for the adaptive MVDR beamformer, implemented as square root information filter is shown in Algorithm 4.

3.4.1 Computational Complexity

The computational complexity of the MVDR square root information filter during training is basically determined by the Householder transformation of the pre-matrix

$$\begin{pmatrix} \mathbf{Q}^{-1/2}(t) & 0 & 0 \\ \hat{\mathbf{S}}_n^{1/2}(t) & \hat{\mathbf{S}}_n^{1/2}(t) & \hat{\mathbf{S}}_n^{1/2}(t)\mathbf{w}_a(t) \\ 0 & \mathbf{R}^{-1/2}(t+1)\mathbf{y}_l(t+1) & \mathbf{R}^{-1/2}(t+1)\mathbf{y}_u(t+1) \end{pmatrix}.$$

A Householder transformation for an $m \times n$ matrix needs $n^2m - \frac{1}{3}n^3$ operations. In case of the MVDR pre-matrix, m equals n and with N microphones, the size of the pre-matrix is equal to:

$$\begin{aligned} m \times n &= ([N-1]+[N-1]+1) \times ([N-1]+[N-1]+1) \\ &= (2N-1) \times (2N-1). \end{aligned}$$

Algorithm 4 Adaptive square root information MVDR beamformer

Known parameters:

- Observation vector $\mathbf{y}_u(t)$
- Upper triangular Cholesky decomposition of inverse process noise covariance matrix at $t = 0$: $\mathbf{Q}^{-1/2}(0)$
- Upper triangular Cholesky decomposition of inverse observation noise covariance matrix at $t = 0$: $\mathbf{R}^{-1/2}(0)$
- Upper triangular Cholesky decomposition of inverse spatial spectral matrix at $t = 0$: $\hat{\mathbf{S}}_n^{-1/2}(0)$
- Initial state estimate $\mathbf{w}_a(0) = \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix}^T$

Computation $t = 1, 2, \ldots$:

- If no speech is present:
  - Fill pre-matrix:

$$\begin{pmatrix} \mathbf{Q}^{-1/2}(t) & 0 & 0 \\ \hat{\mathbf{S}}_n^{-1/2}(t) & \hat{\mathbf{S}}_n^{-1/2}(t) & \hat{\mathbf{S}}_n^{-1/2}(t)\mathbf{w}_a(t) \\ 0 & \mathbf{R}^{-1/2}(t+1)\mathbf{y}_l(t+1) & \mathbf{R}^{-1/2}(t+1)\mathbf{y}_u(t+1) \end{pmatrix}$$

  - Perform Householder transformation on pre-array to get upper triangular post-array:

$$\begin{pmatrix} \mathbf{F}^{-1/2}(t+1) & * & * \\ 0 & \hat{\mathbf{S}}_n^{-1/2}(t+1) & \xi(t+1) \\ 0 & 0 & * \end{pmatrix}$$

  - Calculate new weight estimate with $\hat{\mathbf{S}}_n^{1/2}(t+1)$ and $\xi(t+1)$:

$$\mathbf{w}_a(t+1) = \hat{\mathbf{S}}_n^{-1}(t+1)\xi(t+1)$$

  - Add diagonal loading to $\hat{\mathbf{S}}_n^{1/2}(t+1)$.

This leads to a total number of

$$(2N-1)^3 - \frac{1}{3}(2N-1)^3 = 16N^3 - 24N^2 + 4N - 2$$

operations[2]. We have to perform these calculations for all $2K$ subbands, hence the computational complexity in total is $O(K \cdot N^3)$. The computational complexity to fill the pre-matrix and to obtain $\mathbf{w}_a(t+1)$ can be ignored, as $\mathbf{R}^{-1/2}, \mathbf{y}_l$ and $\mathbf{y}_u$ are scalars and $\hat{\mathbf{S}}_n^{1/2}(t)\mathbf{w}_a(t) = \xi(t)$ which is initialized with zero and in each following step obtained from the post-matrix.

In case of the MVDR beamformer, the computational influence of the cubic complexity is reduced by the fact, that we only have to update the filter (and therefore only have to calculate the Householder transformation) in silence regions. During a talk or meeting, the parts of the signal, which only contain silence will be small compared to the total length of the recording. Still, complexity is much higher compared to the DS beamformer which limits the use of the MVDR beamformer in a real time application to a smaller number of microphones.

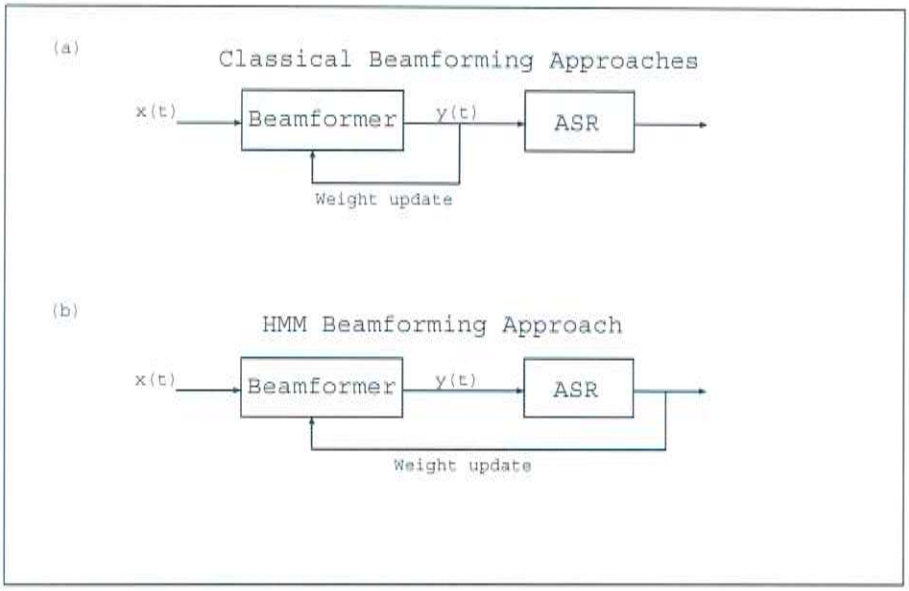
[2] A detailed overview of operations in square root filters is given in [Kaminski71].

Chapter 4

HMM Beamformer

The two beamforming algorithms we presented so far are both trying to maximize the SNR of an incoming signal as shown in Figure 4.1(a). Though both techniques improve the quality of the speech signal, they are not designed to optimize the output of an ASR system. As we want to maximize the performance of an ASR, it would be preferable if we could incorporate its output in the optimization step as shown in Figure 4.1(b). This chapter presents

Figure 4.1: Beamforming concepts



an HMM beamformer, which is designed according to this idea. The beamformer is a further development of M. Seltzers [Seltzer03], J. McDonoughs [McDonough04], and D. Raubs [Raub04] approaches and derived from the ideas in [McDonough05]. Its implementation and evaluation was one of our main goals in this diploma thesis.

Before deriving the HMM beamformer, it is worth summarizing the basic ideas of automatic speech recognition systems to briefly present the main techniques that the HMM beamformer is based on.

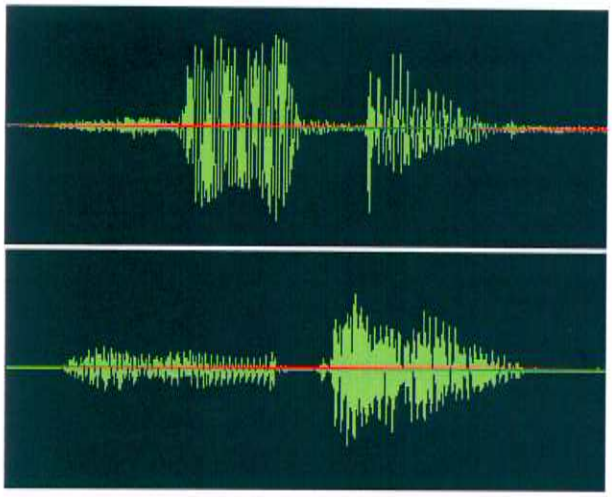
4.1 Automatic Speech Recognition

The HMM beamforming approach presented in this thesis incorporates the output of an ASR system in its adaptation step. It is therefore important to understand the basic concepts of modern speech recognizers. This section will give a short introduction to ASR systems. It will present some basic problems of automatic speech recognition and briefly explain common techniques in modern approaches.

4.1.1 Feature Extraction

Modern ASR systems use statistics and pattern classification to recognize speech [Rabiner93, et al.], as there is no easy way to compare a plain audio signal with another one to find a word or sentence. Figure 4.2 shows two recordings of the same utterance. Though the speaker had tried to reproduce exactly the same words, the length of the recordings and the amplitude differ a lot. Instead of using the plain signal, modern systems extract special features from the recording, which contain all necessary information to do speech recognition and are easily comparable. To extract those features, ASR systems do not use a complete utterance but separate speech into short overlapping frames with an analysis window (e.g. a Hamming window). The length of this window is usually between 10 ms and 25 ms; within this window length, a speech signal is approximately stationary. A feature vector is extracted from these segments and used as an input for the speech recognizer. The next section presents one of these feature vectors called Mel Frequency Cepstral Coefficients and shows how to derive it from a given input signal.

Figure 4.2: Two audio recordings of the same utterance



Mel Frequency Cepstral Coefficients

To extract a feature vector, the speech signal is usually windowed and transformed in the spectral domain by a *discrete Fourier transform* (DFT)

$$\mathbf{X}_t(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \mathbf{x}[n]\mathbf{w}[n-t]e^{-j\omega n} \tag{4.1}$$

where x represents the speech signal, X the signal in the spectral domain and $\mathbf{w}$ a windowing function. Speech is usually neither static nor periodic over a whole utterance but at least static within short segments. Hence, by applying the window function we get static segments which we assume to be periodic. After the Fourier transformation, *Mel warping* is applied by weighting the magnitude squared frequencies with several overlapping triangular functions, the so called *Mel filter*. This can be done by a simple matrix multiplication with the Mel matrix Λ:

$$\hat{\mathbf{X}} = \Lambda \cdot |X|^2 \tag{4.2}$$

Mel-warping usually implies a big dimensional reduction of $\mathbf{X}$: Before Mel-warping $\mathbf{X}$ is a K-dimensional vector, where K equals the FFT filter length (usually in a range between 256 and 2048). After Mel-warping $\hat{\mathbf{X}}$ refers to an M-dimensional vector, where M equals the size of the Mel-matrix and is

usually much smaller than K (e.g. about 30).

By applying *vocal tract length normalization* (VTLN), we can improve speaker independent recognition. Finally, *cepstral features* c can be computed by taking the natural logarithm and doing a *discrete-cosine-transform* (DCT). Given L subband samples and an *inverse discrete Fourier transform* (IDFT) the n-th cepstral coefficient can be expressed by

$$c_n = \frac{1}{2L} \sum_{l=0}^{L-1} \log \tilde{X}_l \cos(\omega_l n). \tag{4.3}$$

Cepstral features give information about the rate of change in the different spectral bands. Mel warped cepstral features (also called *Mel frequency cepstral coefficients* (MFCC)) are widely used in ASR systems [Davis80]. Often, to further reduce the dimension of the feature vector, only the first 13 MFCCs are used. They will be the base for our HMM beamforming approach later on.

4.1.2 Statistical Pattern Recognition

Speech recognizer use statistical methods to do the pattern classification [Rabiner93]. The basic problem of recognizing speech can be described with Bayes' rule:

$$P(c|y) = \frac{P(y|c) \cdot P(c)}{P(y)} \tag{4.4}$$

Given an acoustic input sequence y we look for the most likely class c to which y belongs e.g. the most likely word sequence w for a given audio signal. With (4.4) this corresponds to

$$w = \arg\max_c \left(\frac{P(y|c) \cdot P(c)}{P(y)} \right)$$

As $P(y)$ stays constant when maximizing with respect to c, we can simplify this formula to

$$w = \arg\max_c \left(P(y|c) \cdot P(c) \right) \tag{4.5}$$

From this formula the basic configuration of all speech recognizers can be derived. Modern ASR systems can be roughly divided into three components:

1. An acoustic model $P(y|c)$

2. A language model $P(c)$

3. A decoder

The calculations for the a-posteriori probability in (4.5) are done in the acoustic model. It searches for the most likely input sequence y given the word sequence c. The second probability of (4.5) is represented by the language model. It calculates the likelihood of different word sequences in a language (e.g. the word sequence "How are you?" is more likely than "How house red?"). The decoder combines the input from the acoustic and the language model and calculates the most likely word sequence based on the input sequence and the likelihood of a specific word sequence in this language in general.

In the development of our HMM beamforming algorithm, we will just try to maximize the output of the acoustic model, hence (4.5) can be further simplified to

$$w = \arg\max_{c} \left(P(y|c) \right). \tag{4.6}$$

4.1.3 Hidden Markov Models

From the point of view of ASR systems, words and sentences are random concatenations of phonems[1]. While we are talking, our vocal tract is continuously changing between different states forming lauds. The idea of having several states which are connected by some sort of transition is presented by an HMM. Figure 4.3 shows a set of different HMMs which varies in the complexity and number of allowed transitions. Taking each phonem as a state, we can recognize all existing words by finding the correct connection between possible states. Figure 4.4 shows, how the word "hallo" might be represented by a linear HMM.

A good introduction to HMMs is for example given in [Rabiner86, et al.]. Rabiner states that an HMM is a tuple of five elements:

1. The set of states $S = \{s_1, s_2, ..., s_n\}$.

[1] A phonem is the smallest unit in speech to distinguish the meaning of a word. Changing the phonem in a word either changes the complete word or produces nonsense. There are about 40-50 phonems in English.

Figure 4.3: HMM models

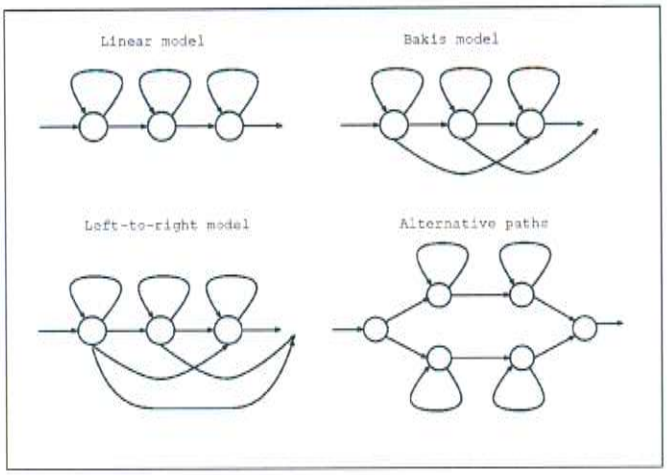
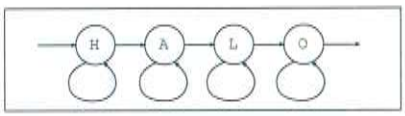


Figure 4.4: HMM model for the word "hallo"



2. The initial state distribution $\pi(s_i)$, $\sum_i \pi(s_i) = 1$, indicating the probability that state s_i is the first state in the Markov chain.

3. The state transition probability distribution $A = \{a_{ij}\}$ where a_{ij} is the probability that state s_j follows state s_i.

4. The discrete or continuous set of possible symbol observations $V = \{v_1, v_2, ..., v_m\}$.

5. The observation symbol probability distribution $B = \{b_j(v)\}$ in state j, where $b_j(v)$ is the probability to observe v while the system is in state s_j.

For our HMM beamforming algorithm, we need an answer to the decoding problem which is given by the *Viterbi alignment* [Viterbi67]. Given an input sentence, the Viterbi algorithm aligns the most likely states with the input sequence. The observations v_m of each state are usually Gaussian mixture

models and stored in a *codebook*. They represent the distribution of feature points over a feature space (in our case, we use MFCC as feature).

Thus, given a correct Viterbi alignment for an input sequence, we can get the associated state to each sample and use it to get the corresponding MFCC means μ and variance Σ from the codebook. We will show, how we can use μ and Σ in our HMM beamformer to update the beam weights in the following section.

4.2 HMM Beamformer Derivation

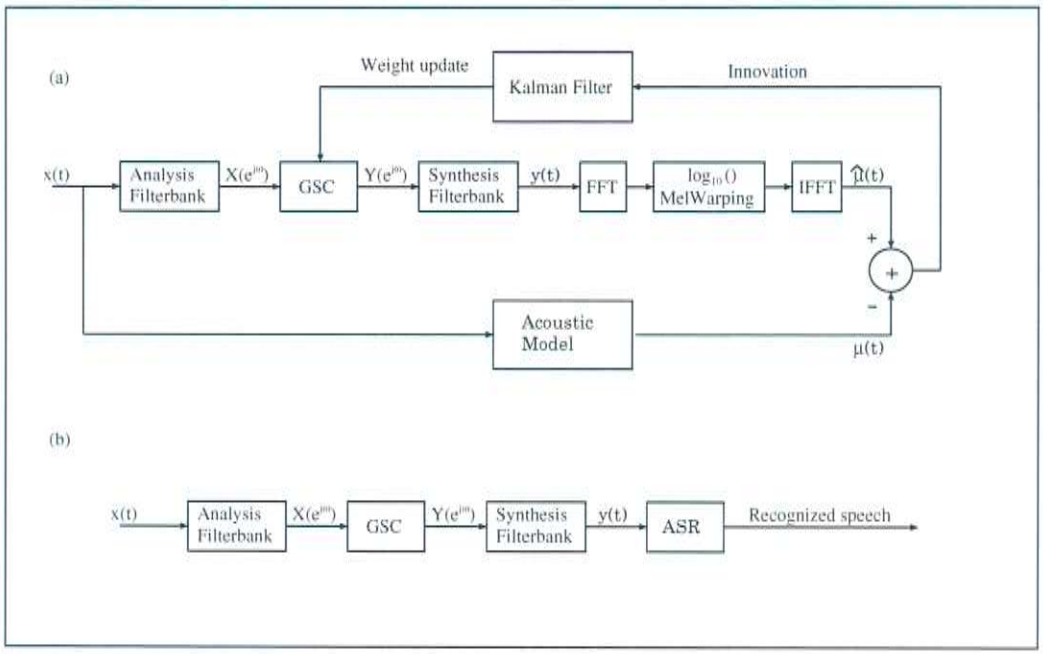
After briefly having summarized some basic concepts of modern speech recognizers in the last section, we now present the derivation of the HMM beamformer based on [Seltzer03]. The HMM beamformer is designed to improve the output of an ASR system by optimizing its beamforming weights with respect to the output of a speech recognizer. While Seltzer used an FFT filterbank and Mel features for his beamforming approach, this new algorithm uses a cosine modulated filterbank and MFCC for the optimization of the beam weights. The FFT filterbank uses overlapping filters which can lead to distortions in the signal. In difference to this the cosine modulated filterbank allows the perfect reconstruction of a signal without aliasing of distortion, as described in Appendix A.

During the optimization, a known input signal $x(t)$ is processed by the HMM beamformer in GSC configuration and an ASR system. The output of the beamformer is used to calculate MFCC. These features are compared with the MFCC of the ASR system, which we get from a Viterbi alignment. The weighted difference $\Delta MFCC$ is eventually used as observation term in a Kalman filter to estimate a weight update for the active weights of the GSC. Figure 4.5 (a) shows the processing chain in the training step. After adapting the beamforming weights, we can use the beamformer to pre-filter the signal before performing speech recognition (see Figure 4.5 (b)).

4.2.1 Formulating the HMM Beamformer as GSC

Similar to the MVDR beamformer, we want to formulate the HMM beamformer as a generalized sidelobe canceler. Let us define snapshot $\mathbf{X}(t)$, as an $N \times 1$ vector, containing the array input for one subband at time t and the active weight $\mathbf{W}(t)$ as an $(N-1) \times 1$ vector, which contains the weights for

Figure 4.5: HMM beamforming configuration for training and speech recognition



$N-1$ microphones for one subband.

In all following equations we present the calculations for one subband only. It is important to realize though, that all calculations have to be done $2K$ times in a real application, similar to the other beamforming algorithms. For example, if the equation states

$$\mathbf{Y}(t) = \mathbf{X}^H(t)\mathbf{B}(t)\mathbf{W}(t)$$

with $\mathbf{B}(t)$ of size $N \times (N-1)$, the correct calculations are given by

$$\mathbf{Y}(t) = \begin{bmatrix} Y_0(t) \\ Y_1(t) \\ \vdots \\ Y_{2K-1}(t) \end{bmatrix},$$

with

$$Y_k(t) = \mathbf{X}_k^H(t)\mathbf{B}(t)\mathbf{W}_k(t),$$

where X_k and W_k represent the k-th snapshot. An explanation why we can treat the different subbands independently is given in Appendix A.

Recall from Section 2.3, that the quiescent weights in the upper branch of a GSC are determined by the distortionless constraint given by (2.6), while the active weights on the lower branch are determined by optimizing a criterion, such as minimizing the total output of the beamformer. Hence, the GSC partitions a constrained optimization problem into a set of constraints and an unconstrained optimization problem.

For the upper branch, we simply multiply the input $\mathbf{X}(t)$ with the quiescent vector $\mathbf{w}_q$

$$Y_u(t) = \mathbf{X}^H(t)\mathbf{w}_q(t). \tag{4.7}$$

When calculating the lower branch, we first have to multiply the input $\mathbf{X}(t)$ with the blocking matrix $\mathbf{B}(t)$, which is orthogonal to $\mathbf{w}_q(t)$, and second with the active weight vector $\mathbf{w}_a$ given by $\mathbf{W}(t)$

$$Y_l(t) = \mathbf{X}^H(t)\mathbf{B}(t)\mathbf{W}(t). \tag{4.8}$$

The output of the beamformer is obtained by subtracting (4.8) from (4.7)

$$Y(t) = Y_u(t) - Y_l(t) = \mathbf{X}^H(t)\left(\mathbf{w}_q(t) - \mathbf{B}(t)\mathbf{W}(t)\right). \tag{4.9}$$

In the end of Section 2.3 we showed, that we can set

$$\mathbf{w}_q(t) = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix},$$

$$\mathbf{B}(t) = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -1 \end{bmatrix},$$

if we pre-steer the input with array manifold vector $\mathbf{v}(k_s)$

$$\tilde{\mathbf{X}}(t) = \mathbf{X}(t) \cdot \mathbf{v}(k_s),$$

where $\cdot$, in this case, indicates an elementwise multiplication.

We now have to find an optimization criterion for $\mathbf{W}(t)$. Our intention is to choose $\mathbf{W}(t)$ so as to maximize the likelihood of an utterance in a training set which is equal to minimizing an auxiliary function.

4.2.2 Gradient Derivation

In [McDonough04] it was demonstrated that the likelihood of a training set $\{\mathbf{c}(t)\}$ can be maximized by minimizing the auxiliary function

$$K = \frac{1}{2} \sum_t \left[\mathbf{c}(t) - \mu(t)\right]^T \mathbf{\Sigma}^{-1}(t) \left[\mathbf{c}(t) - \mu(t)\right], \tag{4.10}$$

where $\mu(t)$ and $\mathbf{\Sigma}(t)$ denote the mean and variance associated with the cepstral feature $\mathbf{c}(t)$. The contribution of one snapshot at time t is then given by

$$\begin{aligned} K(t) &= \frac{1}{2} \left[\mathbf{c}(t) - \mu(t)\right]^T \mathbf{\Sigma}^{-1}(t) \left[\mathbf{c}(t) - \mu(t)\right] \\ &= \frac{1}{2} \left[\mathbf{c}^T(t)\mathbf{\Sigma}^{-1}(t)\mathbf{c}(t) - 2\mathbf{c}^T(t)\mathbf{\Sigma}^{-1}(t)\mu(t) + \mu^T(t)\mathbf{\Sigma}^{-1}\mu(t)\right] \end{aligned} \tag{4.11}$$

To minimize the auxiliary function using the active beam weights $\mathbf{W}(t)$, we want to derive the gradient $\partial K(t)/\partial \mathbf{W}(t)$. This can be done by splitting the formula into three separate pieces. Using the chain rule, we get

$$\frac{\partial K(t)}{\partial \mathbf{W}(t)} = \sum_{n=0}^{J-1} \frac{\partial K(t)}{\partial \mathbf{c} c_n(t)} \cdot \frac{\partial c_n(t)}{\partial \mathbf{W}(t)} = \sum_{n=0}^{J-1} \frac{\partial K(t)}{\partial c_n(t)} \cdot \frac{\partial c_n(t)}{\partial \mathbf{y}(t)} \cdot \frac{\partial \mathbf{y}(t)}{\partial \mathbf{W}(t)}. \tag{4.12}$$

Derivation of $\partial K(t)/\partial c_n(t)$

Taking the partial derivative of (4.11) with respect to $\mathbf{c}(t)$ gives

$$\frac{\partial K(t)}{\partial \mathbf{c}(t)} = \mathbf{\Sigma}^{-1} \left[\mathbf{c}(t) - \mu(t)\right]. \tag{4.13}$$

To reduce complexity, most ASR systems use a diagonal covariance matrix, hence with J cepstral coefficients, $\mathbf{\Sigma}(t)$ can be expressed as

$$\mathbf{\Sigma}^{-1}(t) = \begin{bmatrix} \phi_0(t) & 0 & 0 & \cdots & 0 \\ 0 & \phi_1(t) & 0 & \cdots & 0 \\ 0 & 0 & \phi_2(t) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \phi_{J-1}(t) \end{bmatrix}. \tag{4.14}$$

This allows us to calculate (4.11) with respect to the n-th cepstral coefficient

$$\frac{\partial K(t)}{\partial c_n(t)} = \frac{c_n(t) - \mu_n(t)}{\phi_n(t)},$$

which is the first expression for (4.12):

$$\begin{aligned}\frac{\partial K(t)}{\partial \mathbf{W}(t)} &= \sum_{n=0}^{J-1} \frac{\partial K(t)}{\partial c_n(t)} \cdot \frac{\partial c_n(t)}{\partial \mathbf{y}(t)} \cdot \frac{\partial \mathbf{y}(t)}{\partial \mathbf{W}(t)} \\ &= \sum_{n=0}^{J-1} \frac{c_n(t) - \mu_n(t)}{\phi_n(t)} \cdot \frac{\partial c_n(t)}{\partial \mathbf{y}(t)} \cdot \frac{\partial \mathbf{y}(t)}{\partial \mathbf{W}(t)}. \end{aligned} \tag{4.15}$$

Derivation of $\partial c_n(t)/\mathbf{y}_n(t)$

We now have to find an expression for $\partial \mathbf{c}_n/\partial \mathbf{y}(t)$. Let us repeat the feature extraction process of ASR systems from Section 4.1, equation (4.1) to (4.3). The input signal x is windowed (e.g. with a hamming window $\mathbf{w}_{Ham}$) and Fourier transformed

$$\mathbf{X}_t\left(e^{j\omega}\right) = \sum_{k=-\infty}^{\infty} \mathbf{x}[k]\mathbf{w}_{Ham}[k-t]e^{-j\omega k}. \tag{4.16}$$

Mel-warping and VTLN is usually applied afterwards and done by simple matrix multiplications with Mel-matrix $\mathbf{\Lambda}$ and VTLN-matrix $\mathbf{\Theta}$:

$$\tilde{\mathbf{X}}_t = \Lambda\Theta\mathbf{X}_t$$

The MFCC $\mathbf{c}_t[n]$ associated with $\hat{\mathbf{X}}_t$ are then defined as

$$\mathbf{c}_t[k] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left|\tilde{\mathbf{X}}_\mathbf{t}\left(e^{j\omega}\right)\right|^2 e^{j\omega k}\mathrm{d}\omega \tag{4.17}$$

for all $k = 0, \pm 1, \pm 2, \ldots$. As $\left|\tilde{\mathbf{X}}_t\left(e^{j\omega}\right)\right|$ has even symmetry, we can use Euler's formula to rewrite (4.17) to

$$\mathbf{c}_t[k] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left|\tilde{\mathbf{X}}_t\left(e^{j\omega}\right)\right|^2 \cos(\omega k)\mathrm{d}\omega. \tag{4.18}$$

As $\mathbf{c}_t[k]$ has even symmetry we can reduce calculations to $k = 0, 1, 2, \ldots$.

In our beamforming algorithm, we want to calculate the cepstral coefficients with the beamformed and re-synthesized output $\mathbf{y}(t)$ as shown in Figure 4.5 (a).

Working in the discrete time domain, we can rewrite (4.16)

$$\mathbf{Y}_t\left(e^{j\omega}\right) = \sum_{k=0}^{K} \mathbf{y}[k]\mathbf{w}_{Ham}[k-t]e^{-j\omega k}. \tag{4.19}$$

Mel-warping and VTLN is applied and the square magnitude of the matrix product is calculated:

$$\begin{aligned} \left|\tilde{\mathbf{Y}}\right|^2 &= |\Lambda\Theta Y|^2 \\ &= |\Upsilon \cdot Y|^2, \\ \text{with} \quad \Upsilon &= \Lambda \cdot \Theta, \end{aligned} \tag{4.20}$$

where we suppressed the time index t and $e^{j\omega}$ for convenience. Recall from Section 4.1.1 that Mel-warping usually reduces the dimension of the input vector to M, where M equals the number of Mel features.

Defining the *cosine transformation matrix* $\mathbf{S}$ with components

$$\mathbf{S}_{nm} = \frac{1}{2M}\cos\frac{nm}{2\pi M} \qquad 0 \leq n < J,\ 0 \leq m < M \tag{4.21}$$

we can rewrite (4.18) as

$$c_n = \sum_{m=0}^{M-1} \mathbf{S}_{nm}\log\left|\tilde{Y}\right|^2 \tag{4.22}$$

where again M equals the number of Mel features and J equals the number of cepstral coefficients (usually $J < M$).

Taking a partial derivative on both sides with respect to y gives:

$$\frac{\partial c_n}{\partial \mathbf{y}} = \sum_{m=0}^{M-1} \frac{\mathbf{S}_{nm}}{\left|\tilde{\mathbf{Y}}\right|^2} \cdot \frac{\partial\left|\tilde{\mathbf{Y}}\right|^2}{\partial \mathbf{y}}. \tag{4.23}$$

To calculate $\partial\left|\tilde{\mathbf{Y}}\right|^2/\partial\mathbf{y}$, let us rewrite (4.19) as the product of the windowed signal $\mathbf{y}_w$ with the DFT matrix $\mathbf{W}_{DFT}$

$$\mathbf{Y} = \mathbf{W}_{DFT}^H \mathbf{y}_w.$$

Thus

$$\begin{aligned} |\mathbf{Y}|^2 &= \left|\mathbf{W}_{DFT}^H \mathbf{y}_w\right|^2 = (\mathbf{y}_w)^T \mathbf{W}_{DFT} \mathbf{W}_{DFT}^H \mathbf{y}_w \\ &= \mathbf{y}^T \mathbf{H} \mathbf{y}, \end{aligned} \tag{4.24}$$

where $\mathbf{H}$ denotes the matrix obtained from the outer product

$$\mathbf{H} = (\mathbf{W}_{DFT}\mathbf{w}_{Ham}) (\mathbf{W}_{DFT}\mathbf{w}_{Ham})^H . \tag{4.25}$$

We can now use (4.24) to calculate the partial derivative

$$\begin{aligned} \frac{\partial \left|\tilde{\mathbf{Y}}\right|^2}{\partial \mathbf{y}} &= \frac{\partial \Upsilon |\mathbf{Y}|^2}{\partial \mathbf{y}} = \frac{\Upsilon \partial |\mathbf{Y}|^2}{\partial \mathbf{y}} \\ &= \frac{\Upsilon \partial \mathbf{y}^T \mathbf{H} \mathbf{y}}{\partial \mathbf{y}} = 2\mathbf{y}^T \Upsilon \mathbf{H} && (4.26) \\ &= 2\mathbf{y}^T \mathbf{G}, && (4.27) \\ \text{with} \quad \mathbf{G} &= \Upsilon \mathbf{H}. && (4.28) \end{aligned}$$

Substituting (4.27) into (4.23) gives

$$\begin{aligned} \frac{\partial c_n}{\partial \mathbf{y}} &= 2 \cdot \mathbf{y}^T \sum_{m=0}^{M-1} \frac{\mathbf{S}_{nm} y}{\left|\tilde{\mathbf{Y}}\right|^2} \cdot \mathbf{G}_m && (4.29) \\ &= \mathbf{y}^T \mathbf{L}_n, && (4.30) \end{aligned}$$

where

$$\mathbf{L}_n = 2 \sum_{m=0}^{M-1} \frac{\mathbf{S}_{nm}}{\left|\tilde{\mathbf{Y}}\right|^2} \cdot \mathbf{G}_m \tag{4.31}$$

and $\mathbf{G}_m$ equals the m-th row of $\mathbf{G}$. Note that we can calculate and store $\mathbf{G}$ in advance to save computational time.

Using (4.29) in (4.15) we get

$$\frac{\partial K(t)}{\partial W(t)} = \sum_{n=0}^{J-1} \frac{c_n(t) - \mu_n(t)}{\phi_n(t)} \cdot \mathbf{y}^T(t) \mathbf{L}_n \cdot \frac{\partial \mathbf{y}(t)}{\partial \mathbf{W}(t)}.$$

Finally, we have to derive $\frac{\partial \mathbf{y}(t)}{\partial \mathbf{W}(t)}$ to complete the formulation of (4.12).

Derivation of $\partial \mathbf{y}(t)/\partial \mathbf{W}(t)$

As shown in (4.9), the output of the GSC equals

$$\mathbf{Y}(t) = \mathbf{X}^H(t)\left(\mathbf{w}_q(t) - \mathbf{B}(t)\mathbf{W}(t)\right).$$

To get the audio signal $\mathbf{y}(t)$ we have to resynthesize the subband snapshot $\mathbf{Y}(t)$ with a synthesis filterbank. As explained in Appendix A, the synthesis filterbank is a delayed chain of filters, that can be presented by one polyphase matrix $\mathbf{P}$ of size $2K \times (2L-1)$. The output of the synthesis filter is

$$\mathbf{p}(t) = \sum_{l=0}^{L-1} \mathbf{P}_l \mathbf{Y}(t) = \sum_{l=0}^{L-1} \mathbf{P}_l \left[\mathbf{X}^H(t)\left(\mathbf{w}_q(t) - \mathbf{B}(t)\mathbf{W}(t)\right)\right]. \tag{4.32}$$

It contains one half of the complete output signal $\mathbf{y}(t)$, which can be presented as stacked column signal

$$\mathbf{y}(0) = \begin{pmatrix} \mathbf{p}_{L-1}(0) \\ \vdots \\ \mathbf{p}_0(0) \\ 0 \\ \vdots \\ 0 \end{pmatrix}, y(1) = \begin{pmatrix} \mathbf{p}_{L-1}(1) \\ \vdots \\ \mathbf{p}_0(1) \\ \mathbf{p}_{L-1}(0) \\ \vdots \\ \mathbf{p}_0(0) \end{pmatrix}, y(2) = \begin{pmatrix} \mathbf{p}_{L-1}(2) \\ \vdots \\ \mathbf{p}_0(2) \\ \mathbf{p}_{L-1}(1) \\ \vdots \\ \mathbf{p}_0(1) \end{pmatrix}, \ldots,$$

where $p_l(t)$ indicates the l-th element of $\mathbf{p}$ at time t. Defining

$$\mathbf{p}^{(0)}(t) = \begin{pmatrix} p_{L-1}(t) \\ \vdots \\ p_0(t) \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{and} \quad \mathbf{p}^{(1)}(t) = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ p_{L-1}(t) \\ \vdots \\ p_0(t) \end{pmatrix},$$

the complete output signal $\mathbf{y}(t)$ is given by

$$\mathbf{y}(t) = \mathbf{p}^{(0)}(t) + \mathbf{p}^{(1)}(t-1).$$

Taking partial derivatives on both sides of the last equation with respect to $\mathbf{W}(t)$ gives

$$\frac{\partial \mathbf{y}(t)}{\partial \mathbf{W}(t)} = \frac{\partial \mathbf{p}^{(0)}(t)}{\partial \mathbf{W}(t)} + \frac{\partial \mathbf{p}^{(1)}(t-1)}{\partial \mathbf{W}(t)},$$

where $\partial \mathbf{p}^{(i)}(t)/\partial \mathbf{W}(t)$ is calculated by taking the partial derivative of (4.32) with respect to $\mathbf{W}(t)$

$$\frac{\partial \mathbf{p}(t)}{\partial \mathbf{W}(t)} = \sum_{l=0}^{L-1} \mathbf{P}_l \left[-\mathbf{X}^H(t)\mathbf{B}(t) \right].$$

With this derivation, we are finally able to solve (4.12)

$$\begin{aligned} \frac{\partial K(t)}{\partial \mathbf{W}(t)} &= \sum_{n=0}^{J-1} \frac{\partial K(t)}{\partial c_n(t)} \cdot \frac{\partial c_n(t)}{\partial \mathbf{y}(t)} \cdot \frac{\partial \mathbf{y}(t)}{\partial \mathbf{W}(t)} \\ &= \sum_{n=0}^{J-1} \frac{c_n(t) - \mu_n(t)}{\phi_n(t)} \mathbf{y}^T(t) \mathbf{L}_n \left[\frac{\partial \mathbf{p}^{(0)}(t)}{\partial \mathbf{W}(t)} + \frac{\partial \mathbf{p}^{(1)}(t-1)}{\partial \mathbf{W}(t)} \right]. \end{aligned} \quad (4.33)$$

We can now use this gradient to optimize the HMM beamforming weights.

4.3 Optimizing HMM Beamforming Weights

The difficulties to optimize the beamforming weights depend strongly on the available information about the speech data. In the best case, we have hand labeled training utterances to calculate the gradient, in the next best case, we have a known training sentences but the Viterbi alignment is done by the speech recognizer, in the worst case, there is no training data and we have to use an ASR system to produce MFCC on the fly. In the first case we talk of *oracle* data, in the second one of *calibrated* data and in the last case of *unsupervised* data. If the entire training utterance is available, we can use a *global optimization* procedure based on a gradient of the whole utterance, if the training data is only available sample by sample, we need an *iterative optimization* algorithm.

As the quality of the beamforming weight training with unsupervised and calibrated data is highly depending on the quality of the speech recognizer, we will focus on oracle data in our future discussion. Thus, we assume that for each sample in the training data, the correct cepstral mean μ and variance $\mathbf{\Sigma}$ are given by the ASR system.

4.3.1 Global Optimization Algorithms

The least complex case arises when a complete utterance is available during the training. To update the beam weights, we first calculate the sum of the

auxiliary function

$$K = \frac{1}{2} \sum_t \left[\mathbf{c}(t) - \mu(t)\right]^T \mathbf{\Sigma}^{-1}(t) \left[\mathbf{c}(t) - \mu(t)\right]. \tag{4.34}$$

Using the gradient $\frac{\partial K}{\partial \mathbf{W}}$, we then adjust the active weights of the beamformer to minimize K. Several algorithms exist which require only the gradient for given beam weights and a stopping criterion to run the optimization automatically. In our global optimization experiments we used a modified version of the "Multimin" package of the GNU Scientific Library, which is a conjugate gradient search, to perform these optimization.

We still have to take care of the problems with array perturbations and mismatches between the estimated and the actual position of the speaker as described in Section 2.7. Without diagonal loading, the global optimization algorithm suffers from the same problems as the MVDR beamformer: The beam weights may grow extremely large and the algorithm becomes unstable.

As the global optimization algorithm offers no easy way to directly implement diagonal loading, we simulate a quadratic constrain by adding a penalty function to (4.34):

$$\hat{K} = K + \alpha P,$$

with penalty factor α and penalty term

$$P = a_{samp} \cdot \mathbf{w}_a^H \mathbf{w}_a,$$

where a_{samp} is a constant factor, depending on the number of samples in the relevant utterance. This penalty function will grow quadratically with growing beam weights. As the global optimization algorithm tries to minimize the auxiliary function (4.34), it will therefore try to keep the magnitude of the beam weights below a certain level, depending on penalty factor α.

4.3.2 Iterative Optimization Algorithms

If we don't have access to the complete utterance for example because we want to train the beam weights on the fly, while recording a speaker, we cannot use a global optimization algorithm. Instead of using the complete sum in (4.34) we can just calculate the weighted difference between our calculated cepstral feature and the cepstral means for one time step:

$$K(t) = \frac{1}{2} \left[\mathbf{c}(t) - \mu(t)\right]^T \mathbf{\Sigma}^{-1}(t) \left[\mathbf{c}(t) - \mu(t)\right]. \tag{4.35}$$

With (4.33) it is straightforward to derive an LMS optimization equation

$$\begin{aligned}\mathbf{W}(t+1) &= \mathbf{W}(t) - \alpha(t)\left[\frac{\partial K(t)}{\partial \mathbf{W}(t)}\right]^H \\ &= \mathbf{W}(t) - \alpha(t)\left[\sum_{n=0}^{J-1} \frac{c_n(t) - \mu_n(t)}{\phi_n(t)} \cdot \frac{\partial c_n(t)}{\partial \mathbf{W}(t)}\right]^H,\end{aligned}$$

where $\alpha(t)$ is a time dependent step size. Unfortunately, the step size of an LMS algorithm is restricted to

$$0 < \alpha(t) < \frac{1}{\lambda_{max}},$$

where λ_{max} equals the maximum eigenvalue of the gradient. We want to avoid this restriction and therefore formulate the iterative HMM beam weight optimization similar to the adaptive MVDR beamformer as square root information filter.

4.3.3 HMM Beamformer as Square Root Information Filter

The goal of our beamforming algorithm is the optimization of the beamforming weights by minimizing (4.35). Minimizing this function is a least mean square problem, which can be solved by a Kalman filter. We want to solve the optimization problem in a square root information filter implementation, similar to the one we used for the adaptive MVDR beamformer.

To formulate the HMM beamformer as square root information filter, we have to find an appropriate correspondence between the square root information filter variables and those, given by the HMM beamformer. The necessary variables for the square root algorithms are:

- the estimated state $\hat{\mathbf{X}}(t)$,
- the covariance matrix of the state estimation error $\mathbf{K}(t)$,
- the state transition matrix $\mathbf{A}(t)$,
- the observation matrix $\mathbf{C}(t)$,

- the correlation matrices of the process- and measurement noise $\mathbf{Q}(t)$ and $\mathbf{R}(t)$ and
- the observation $\mathbf{Y}(t)$.

As the HMM beamformer is formulated as GSC like the MVDR beamformer, some of the correspondences from Section 3.4 can be directly transferred: The state estimation $\hat{\mathbf{X}}(t)$ is given by the active weight vector $\mathbf{W}(t)$, the state transition matrix can be set to identity $\mathbf{A}(t) = \mathbf{I}$ and an appropriate value for the correlation matrix of process noise $\mathbf{Q}(t)$ has to be determined experimentally. $\mathbf{K}(t)$ can be used to apply diagonal loading. Similar to equation (3.15) of the MVDR beamformer, we just have to construct one pre-matrix with $K(t)$ and the diagonal loading matrix $\sigma^2(t)\mathbf{I}$ and transfer it into a post-matrix by using the Householder transformation:

$$\mathbf{U}_2 \begin{bmatrix} \mathbf{K}^{-1/2}(t+1) \\ ------ \\ \sigma^2(t)\mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_\sigma^{-1/2}(t+1) \\ ------ \\ 0 \end{bmatrix}. \tag{4.36}$$

We still have to find corresponding values for $\mathbf{Y}(t)$, $\mathbf{C}(t)$ and $\mathbf{R}(t)$, though. Let us assume that our observation is an MFCC vector $\mathbf{c}(t)$. Then $C(t)$ must be able to transfer the weight vector $W(t)$ into a predicted observation $\hat{c}(t+1|t)$ as in the Kalman filter equation (3.3). This can be done by the gradient

$$\begin{aligned} \frac{\partial c(t)}{\partial \mathbf{W}(\mathbf{t})} &= \sum_{n=0}^{J-1} \frac{\partial c_n(t)}{\partial y(t)} \cdot \frac{\partial y(t)}{\partial \mathbf{W}(t)} \\ &= \sum_{n=0}^{J-1} y^T(t)\mathbf{L}_n \cdot \left[\frac{\partial p^{(0)}(t)}{\partial \mathbf{W}(t)} + \frac{\partial p^{(1)}(t-1)}{\partial \mathbf{W}(t)} \right] \end{aligned} \tag{4.37}$$

which was one part of our gradient derivation in Section 4.2.2.

The next observation $\mathbf{c}(t+1|t+1)$ should be equal to $\hat{\mathbf{c}}(t+1|t)$ plus some small difference β:

$$\mathbf{c}(t+1|t+1) = \hat{\mathbf{c}}(t+1|t) + \beta(t).$$

This difference is directly observable as the difference between the calculated MFCC $c(t+1)$ from the beamformed signal and the MFCC mean vector μ

Table 4.1: Correspondences between sqrt. info. filter and HMM beamformer

SqrtInfoFilter	HMM beamformer
X(t)	$\mathbf{W}(t)$
Y(t)	$\hat{c}(t\|t-1) + [c(t) - \mu(t)]$
C(t)	$\partial \mathbf{c}(t) / \partial \mathbf{W}(t)$
$A(t)$	$\mathbf{I}$
$Q(t)$	$\sigma \mathbf{I}$
$R(t)$	Σ

from the ASR. Thus, the new observation $\mathbf{Y}(t+1) = \mathbf{c}(t+1|t+1)$ is given by

$$\begin{aligned} c(t+1|t+1) &= \hat{\mathbf{c}}(t+1|t) + \beta(t+1) \\ &= \hat{\mathbf{c}}(t+1|t) + [\mathbf{c}(t+1) - \mu(t+1)], \end{aligned} \tag{4.38}$$

where $c(t+1)$ is calculated according to (4.22).

Using the MFCC $c(t|t)$ as observation indicates the use of the MFCC variance $\mathbf{\Sigma}$ from the ASR as measurement noise covariance matrix, hence, all necessary correspondences were found to use the HMM beamformer in a square root information filter implementation. Table 4.1 shows all corresponding variables.

Given these correspondences and the square root information filter pre-matrix

$$\mathbf{\Gamma} = \begin{pmatrix} \mathbf{Q}^{-1/2}(t) & 0 & 0 \\ \mathbf{K}^{-1/2}(t)\mathbf{A}^{-1}(t) & \mathbf{K}^{-1/2}(t) & \mathbf{K}^{-1/2}(t)\hat{\mathbf{X}}(t) \\ 0 & \mathbf{R}^{-1/2}(t+1)\mathbf{C}(t+1) & \mathbf{R}^{-1/2}(t+1)\mathbf{Y}(t+1) \end{pmatrix},$$

it is straight forward to implement the training of an HMM beamformer as square root information filter, as shown in Algorithm 5. You may notice, that it is very similar to Algorithm 4, which allows us to use the same algorithms for the Householder transformation and diagonal loading.

4.3.4 Computational Complexity

The computational complexity for the HMM square root information filter is primarily determined by two calculations: The gradient given by (4.37) and the Householder transformation of the pre-matrix Γ.

Algorithm 5 HMM square root information filter

- Initialize beamforming weights $\mathbf{W}(0) = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$
- Initialize upper triangular Cholesky decomposition of inverse spacial spectral matrix at $t = 0$: $\mathbf{K}^{-1/2}(0) = \sigma_S \cdot \mathbf{I}$
- Initialize upper triangular Cholesky decomposition of inverse process noise covariance matrix : $\mathbf{Q}^{-1/2}(t) = \sigma_R \cdot \mathbf{I}$

Computation $t = 1, 2, \ldots$:

- Estimate $\mathbf{c}(t)$ from the output of the beamforming process chain and calculate $\frac{\partial \mathbf{c}}{\partial \mathbf{W}(t)}$.
- Get $\mu(t)$ and $\mathbf{\Sigma}(t)$ from the acoustic model of an ASR for the given sample.
- Use $\mathbf{W}(t)$ and $\partial \mathbf{c}/\partial \mathbf{W}(t)$ to calculate $\hat{\mathbf{c}}(t+1|t)$.
- Calculate $\mathbf{c}(t+1|t+1) = \hat{\mathbf{c}}(t+1|t) + [\mathbf{c}(t+1) - \mu(t+1)]$
- Fill pre-matrix:

$$\begin{pmatrix} \mathbf{Q}^{-1/2}(t) & 0 & 0 \\ \mathbf{K}^{-1/2}(t) & \mathbf{K}^{-1/2}(t) & \mathbf{K}^{-1/2}(t)\mathbf{W}(t) \\ 0 & \mathbf{\Sigma}^{-1/2}(t+1)\mathbf{C}(t+1) & \mathbf{\Sigma}^{-1/2}(t+1)\mathbf{c}(t+1|t+1) \end{pmatrix}$$

- Perform Householder transformation on pre-array to get upper triangular post-array:

$$\begin{pmatrix} \mathbf{F}^{-1/2}(t+1) & * & * \\ 0 & \mathbf{K}^{-1/2}(t+1) & \xi(t+1) \\ 0 & 0 & * \end{pmatrix}$$

- Calculate new weight estimate with $\mathbf{K}^{-1/2}(t+1)$ and$\xi(t+1)$:

$$\mathbf{W}(t+1) = \mathbf{K}(t+1)\xi(t+1)$$

- Add diagonal loading to $\mathbf{K}^{-1/2}(t+1)$.

Let us rewrite (4.37):

$$\frac{\partial \mathbf{c}(t)}{\partial \mathbf{W}(t)} = \mathbf{y}^T(t)\mathbf{L}_n \left[\frac{\partial \mathbf{p}^{(0)}(t)}{\partial \mathbf{W}(t)} + \frac{\partial \mathbf{p}^{(1)}(t-1)}{\partial \mathbf{W}(t)} \right],$$

with

$$\mathbf{L}_n = 2 \sum_{m=0}^{M-1} \frac{\mathbf{S}_{nm}}{\left| \tilde{\mathbf{Y}} \right|^2} \mathbf{G}_m$$

and

$$\frac{\partial p^{(i)}(t)}{\partial \mathbf{W}(t)} = \sum_{L=0}^{L-1} P_l \left[-\mathbf{X}^H(t)\mathbf{B}(t) \right]. \tag{4.39}$$

As stated, we can calculate $\mathbf{G}_m$ in advance; hence, the computational time to calculate $\mathbf{L}_n$ is negligible. Using the fact, that

$$\mathbf{B}(t) = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -1 \end{bmatrix},$$

we can calculate the vector-matrix product $\mathbf{X}^H(t)\mathbf{B}(t)$ in (4.39) with

$$\mathbf{X}^H\mathbf{B} = \begin{bmatrix} \mathbf{X}[0] - \mathbf{X}[1] \\ \mathbf{X}[1] - \mathbf{X}[2] \\ \vdots \\ \mathbf{X}[N-2] - \mathbf{X}[N-1] \end{bmatrix}, \tag{4.40}$$

where $\mathbf{X}[i]$ indicates the i-th element of vector $\mathbf{X}$. Thus, the computational complexity of (4.37) is determined by the vector product and sum of $\mathbf{P}_l$ and (4.40), where $\mathbf{P}_l$ is a vector of size $2K \times 1$. As we have to compute this gradient for each of the $2K$ subbands, the total computational complexity is given by $O\left(L \cdot K^2 \cdot (N-1)\right)$.

To estimate the computational complexity of the Householder transformation in Section 3.4.1, we found that one Householder transformation needs approximately $n^2 m - \frac{1}{3}n^3$ operations for an $m \times n$ matrix. For the HMM square root information filter, the pre-matrix is given by

$$\begin{pmatrix} \mathbf{Q}^{-1/2}(t) & 0 & 0 \\ \mathbf{K}^{-1/2}(t) & \mathbf{K}^{-1/2}(t) & \mathbf{K}^{-1/2}(t)\mathbf{W}(t) \\ 0 & \mathbf{\Sigma}^{-1/2}(t+1)\mathbf{C}(t+1) & \mathbf{\Sigma}^{-1/2}(t+1)\mathbf{c}(t+1|t+1) \end{pmatrix},$$

with sizes of

$$\begin{aligned}
\mathbf{Q}^{-1/2}(t) &= N \times N \\
\mathbf{K}^{-1/2}(t) &= (N-1) \times (N-1) \\
\mathbf{W}_k(t) &= (N-1) \times 1 \\
\mathbf{\Sigma}^{-1/2}(t) &= N_{cep} \times N_{cep} \\
\mathbf{C}(t) &= N_{cep} \times N \\
\mathbf{c}(t|t) &= N_{cep} \times 1,
\end{aligned}$$

where N_{cep} equals the number of MFCC. Thus, the size of the complete pre-matrix is

$$(N + (N-1) + N_{cep}) \times (N + (N-1) + 1) = (2N + N_{cep} - 1) \times (2N)$$

and the complete number of operations to perform the Householder transformation is given by

$$(2N)^2 \cdot (2N + N_{cep} - 1) - \frac{1}{3}(2N)^3 = 8N^3 + 2N^2 N_{cep} - 4N^2 - \frac{8}{3}N^3$$

We have to perform this transformation for all $2K$ sub samples. Hence the complete complexity factor is $O(K \cdot N^3)$. A detailed overview over the necessary operations for a square root information filter using the Householder transformation is given in [Kaminski71].

To fill the pre-matrix, additional operations are necessary to calculate the matrix product $\Sigma^{-1/2}(t+1)C(t+1)$ and $\Sigma^{-1/2}(t+1)\mathbf{c}(t+1|t+1)$. Fortunately, $\Sigma^{-1/2}$ is a diagonal matrix, hence the matrix-matrix product $\Sigma^{-1/2}(t+1)C(t+1)$ can be computed in $O(K \cdot N_{cep} \cdot N)$ and the matrix-vector product $\Sigma^{-1/2}(t+1)\mathbf{c}(t+1|t+1)$ in $O(K \cdot N_{cep})$.

Overall, we can state that this algorithm is by far the most complex approach. Though the computational complexity for the Householder transformation seems to be equal to the MVDR algorithm, we have to remember, that the MVDR beamformer only updates the beam weights, when no speech is present, while the HMM beamformer uses the complete recording to update the beam weights. Another important impact on the computational time is given by the gradient calculation (4.37). As $K >> N$ in most applications, the gradient calculations won't get much faster even with a reduced number of microphones in the array. Thus, the HMM beamformer will hardly run in realtime even with a low number of microphones.

Chapter 5

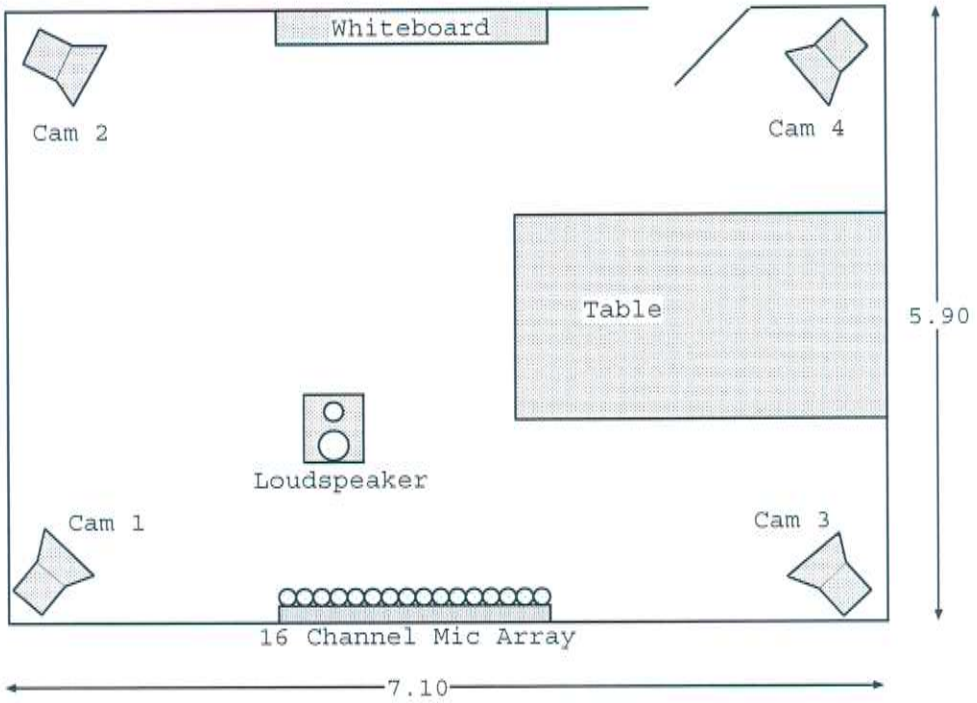
Experimental Configuration and Results

To evaluate the performance of the three different beamforming approaches in the context of ASR, we used the same configuration as in [McDonough04]. The Millennium automatic speech recognition system, a state of the art speech recognizer, scored the different beamformers. It is developed and maintained by the Interactive Systems Laboratories at University Karlsruhe, Germany. For our experiments, the Millennium used an unadapted HMM with 48 Gaussians for about 2000 codebooks. It was trained on the English Spontaneous Scheduling Task (ESST) corpus.

This speech corpus contains approximately 35 hours of speech contributed by 242 speakers. The recordings consist of speech dialogs between two speakers, talking about oversea business trips and their organisation. Recorded with Sennheiser head-mounted, close-talking microphones, this clean speech was used to train the HMM speech recognizer for all experiments.

From the ESST corpus, a *test set* was extracted and used for our beamforming experiments. It is approximately 3.5 hours in length and contains 58 dialog halves contributed by 16 unique speakers with at total number of 22,889 words. The length of the utterances varies from a few seconds to approximately 1.5 minutes. To obtain far field recordings from the original close-talking recordings, the clean speech test data was played through a loudspeaker into a 6×8 meter room with a reverberation time of approximately 350 ms, and then recorded with a linear microphone array. The array consisted of 16 Sennheiser omnidirectional microphones with equidistant spacing of 4.1 cm. To reduce computational complexity, only the first eight micro-

Figure 5.1: ESST room configuration



phones of the 16 channel microphone array were used in our beamforming experiments. The stationary loudspeaker was located 2 m in front of one end of the array. In addition to the source speech data, interference data was also recorded by moving the speaker parallel to the array approximately 3 m from the location used for the source. All speech data was digitally sampled with 16 kHz. The configuration of the room is shown in Figure 5.1.

We implemented all beamforming algorithms for the experiments in C++ and Python. While we used Python primarily to to load and store the data files, we implemented most time consuming calculations in C++ making heavily use of the numerical implementations in the GNU Scientific Library.

5.1 Experiments without Interference

In the first experiments, we used the ESST test set without interference to evaluate the beamformers in a controlled environment without a moving speaker or cross talk from other speakers. Noise and disturbances from com-

puter fans and reverberation in the far field recordings are low. To show the general impact of beamforming to a far field signal, only the first microphone in the array was used to generated an ASR hypothesis without beamforming. This lead to a high WER of 63.83% for the adapted decoding step. Compared to the good performance of the Millennium on a close talking microphone with only 31.94% WER, this result indicates the big influence of noise and disturbances to ASR even under good acoustic conditions.

5.1.1 DS Beamforming Experiments

Using a DS beamformer as described in Section 2.1, we could dramatically improve the WER to 58.35 percent for the adapted decoding run. We use this result as a baseline for our adaptive beamforming algorithms. With eight microphones, beamforming was done in almost two hours on an Intel Xeon 3 GHz. This is faster than realtime and shows the good performance of the algorithm. If we take into account that we have to subtract some time for the I/O operations of the system to get the speech data and store the wave files, there is no doubt that we can easily increase the number of microphones by a factor of two to four and still be able to process data in real time.

5.1.2 MVDR Beamforming Experiments

The MVDR beamformer was evaluated in a square root information filter implementation as described in Section 3.4. The longest utterance in each of the 58 dialogs was used to adapt the beam weights in the training step. In parallel to the beamforming input, *oracle* state alignments were used to detect speech pauses; beam weights were only updated when the Viterbi labels of the Viterbi alignment indicated a silent region. The Viterbi alignment was obtained using the correct transcription on the output of the DS beamformer. After training the beamformer, the adapted weights were used to process all utterances of the test set.

We tried different levels of diagonal loading to increase the stability of the beamformer. The range for diagonal loading is limited by two facts: If we increase the diagonal loading too much, the algorithm is unable to adapt. In the worst case, beam weights are limited to zero and we obtain a DS beamformer as described in Section 2.7. On the other hand, if we decrease the diagonal loading too much, the beamweights may grow too much and the beamformer would become unstable.

Table 5.1: Results for MVDR beamformer on ESST test set

Diagonal loading	% WER unadapted	% WER adapted
-25 dB	63.12	59.04
-30 dB	61.47	**57.83**
-35 dB	61.33	58.35
DS beamformer	61.27	58.35

Our best result was obtained with a diagonal loading of -30 dB and produced a WER of 57.83%. This is an absolute difference of 0.52% compared to the DS baseline. Table 5.1 shows the complete test set for the MVDR beamformer with different diagonal loading.

With eight microphones, the algorithm took about 3 hours on an Intel Xeon 3 GHz to train the beam weights. Given that the longest utterances in each of the 58 dialogues had an approximated length of one minute, this is about 3 times real time.

5.1.3 HMM Beamforming Experiments

Similar to [McDonough04] we used *oracle* state alignments for all HMM beamforming experiments. The state alignment was obtained by a Viterbi algorithm with correct transcription on the output of a DS beamformer. During the training of the beamforming weights, we only used a single Gaussian component per codebook and only static cepstral features for the weight optimization.

To calculate the MFCC with 13 cepstral components, we used the output of the synthesis filterbank as described in Section 4.2.2. Features were calculated every 10 ms using a 20 ms sliding window. A Hamming window with a length of 320 samples was applied to each segment of speech. The windowed segment was padded with zeros to a length of 512 and an FFT was calculated. VTLN and mel warping with a mel matrix of size 30×512 was applied to the sample as described in (4.20). The cepstral features were obtained by calculating the logarithm of the squared magnitude of the sample and performing a cosine transformation as shown in (4.22).

Global Optimization Algorithm

In our introduction to HMM beamformers, we distinguished two different situations: In Section 4.3.1, we assumed, that the complete utterance is available, while Section 4.3.3 is based on the assumption, that we have to train the beamformer iteratively.

With respect to the first assumptions, we used a global optimization algorithm to adapt the beam weights. Similar to the MVDR experiment, only the longest utterance of each dialog of the ESST test set was taken to estimate a gradient and a likelihood value for a certain beamformer configuration. We used an adapted version of the GNU Scientific Library "Multimin" package for the optimization. It takes the gradient and performs several gradient descent steps to find a local minimum, then re-estimates the gradient at this minimum to perform another gradient descent search. The optimization stops after a given number of iterations or after the improvement is below a given threshold. We limited the number of iterations to five and the threshold to $1.0E^{-04}$. In most cases, the threshold was reached before the maximum number of iterations.

Using the global optimization algorithms, we cannot easily apply diagonal loading during the update procedure, but without limitations to the beam weights, the gradient descent algorithm becomes unstable. Section 4.3.1 presented another solution: Instead of using diagonal loading, a penalty factor is added to the auxiliary function (4.34), which stabilizes the optimization algorithm.

To show the influence of the penalty term, we evaluated the HMM beamformer with a penalty factor of $\alpha = 1.0E^{-5}, \alpha = 1.0E^{-6}$, $\alpha = 1.0E^{-7}$ and $\alpha = 0.0$. With a penalty factor of zero and $1.0E^{-7}$ the weights grew above all reasonable values, resulting in a low performance of the ASR system. On the other hand, if the penalty factor is too high, beam weights close to zero are preferred. In the worst case, $w_a^H = [0 \ldots 0]$ and the resulting beamformer is identical to a DS beamformer. The best results were obtained, using a penalty factor of $\alpha = 1.0E^{-5}$ with a WER of 57.98 %, which is 0.37 points better than the DS baseline. Table 5.2 shows the experimental results for the HMM global optimization beamformer.

The computational time to perform the global optimization is depending on how many iterations the algorithm needs to find an optimum for the beamforming weights. The more iterations it takes, the more calculations of a gradient are necessary. In average, a complete experiment took about

Table 5.2: Results of the HMM global optimization on the ESST set

Penalty factor	% WER unadapted	% WER adapted
$\alpha = 0.0$	71.70	70.00
$\alpha = 1.0E^{-7}$	63.44	60.58
$\alpha = 1.0E^{-6}$	61.05	58.23
$\alpha = 1.0E^{-5}$	61.31	**57.98**
$\alpha = 1.0E^{-4}$	61.27	58.35
DS beamformer	61.27	58.35

six days of processing time on 5 Intel Xeon 3 GHz machines to finish. This excludes the algorithm from any application, where real time processing is necessary.

HMM Beamformer in Square Root Information Filter Implementation

In this experiment we used the square root information filter implementation as described in Section 4.3.3. It assumes that the input signal is arriving sample by sample and uses an iterative optimization approach to train the beam weights. Again, training was only performed on the longest utterance of each group in the ESST test set, while all utterances in the ESST test set were used afterwards to evaluate the beamformer. Diagonal loading of -20 dB, -25 dB and -30 dB was applied during the experiments. Unfortunately, there seemed to be a bug in the implementation of the diagonal loading, which could not be solved in the time remaining for the thesis. The bug prevented a correct addition of the diagonal loading after multiple iterations. Without diagonal loading, a WER of 58.35% was obtained, which is equal to the performance of the DS beamformer. The computational time for one experiment is approximately comparable to the global optimization scenario; using five Intel Xeon 3 GHz PCs, one experiment took about five days to complete.

Comparing the best results from the different experiments, we obtained the overall best result with the MVDR beamformer with -30 dB diagonal loading. For the adapted decoding, we get a WER of 57.83%, which is 0.52% absolute better than the DS- and iterative HMM beamformer and 0.15 points better than the global optimization HMM beamformer. Table 5.3 shows the

Table 5.3: Best Results of the Different Beamformers

Beamformer	% WER unadapted	% WER adapted
DS	61.27	58.35
MVDR	61.47	**57.83**
HMM global opt.	61.31	57.98
HMM sqrt. info.	61.76	58.35

Table 5.4: Beamformers with new ASR Decoder

	% WER standard decoder	% WER improved Decoder	Absolute difference
D&S beamformer	58.35	53.65	4.70
MVDR beamformer	57.83	**53.45**	4.38
HMM global opt.	57.98	53.62	4.36
HMM sqrt info.	58.35	53.55	4.80

best results for all beamformers.

5.1.4 Evaluation of the ASR Influence to the Beamforming Results

To evaluate the influence of the ASR decoder to the experimental result, we performed a second decoding run, increasing the beam width and retraining the codebook using VTLN during training. This improved the WER for all beamformer but also reduced the differences between all algorithms, as the performance of the DS- and iterative HMM beamformer could improve by 4.70% and 4.80% absolute, while the MVDR- and HMM global optimization algorithm only gained 4.38% and 4.36% absolute. The MVDR still outperforms all other beamformers with a WER of 53.45% but the difference to the next best system is only 0.1 points. Table 5.4 shows the results for the different beamformers and ASR systems.

As the HMM beamformer is using the feedback of an ASR system to train its beam weights, we performed another experiment, addressing this special issue. During the training step of the HMM beamformer, updates of the beam weights are highly depending on the cepstral means μ obtained from the ASR system. We usually get those means from a Viterbi alignment.

Table 5.5: HMM beamformer with adaptive cepstral means

Beamformer	% WER unadapted	% WER adapted	Improved decoder
HMM	61.31	57.98	53.62
HMM adaptive cepstral means	**61.20**	**58.43**	**53.51**
MVDR	61.47	57.83	53.45
DS	61.27	58.35	53.65

To test the influence of a good Viterbi alignment to the experimental result, we used the MFCC produced in the beamforming process chain, to estimate cepstral means on the fly. As this experiment needs the complete utterance, we could only use the global optimization algorithm.

Taking our best setting with penalty factor $\alpha = 1.0E^{-5}$, we obtained a WER of 61.2% for the unadapted decoding, compared to 61.31% without adapted cepstral means. This is 0.27 points better then the best MVDR beamformer. For the adapted decoding, we obtained a WER of 58.43% which is 0.45% higher than the results obtained when using the Viterbi alignment from the beamformer. Using the new decoder settings for the last experiment, the WER of the adapted decoding improved from 58.43% to 53.51%; only 0.06 points worse than the best MVDR result which is a negligible difference. While the adaptive cepstral means did worse on the old decoder than the Viterbi alignment, they lead to a better performance when choosing the new one. The relevant numbers are shown in Table 5.5.

5.2 Experiments on the ESST Test Set with Additional Interference

All improvements obtained by the different beamforming algorithms in the previous experiments were based on their ability to reduce the influence of noise and reverberation in the room. In a final experiment, we evaluated the performance of the different beamformers on the ESST set with an additional interferer. To simulate this interference, the loudspeaker was moved approximately three meters in parallel to the microphone array and used to playback another talk as interfering cross talk in the backround. As the HMM and MVDR beamforming algorithms are especially designed to adapt

Table 5.6: ESST test set with additive interference

Beamformer	% WER Standard Decoder	% WER Improved Decoder
DS beamformer	63.74	58.36
HMM, adaptive cepstral means	**62.62**	**56.89**
MVDR	64.17	58.26

to an interfering source, we would expect the differences in performance between those algorithms and the DS beamformer to increase under these new conditions.

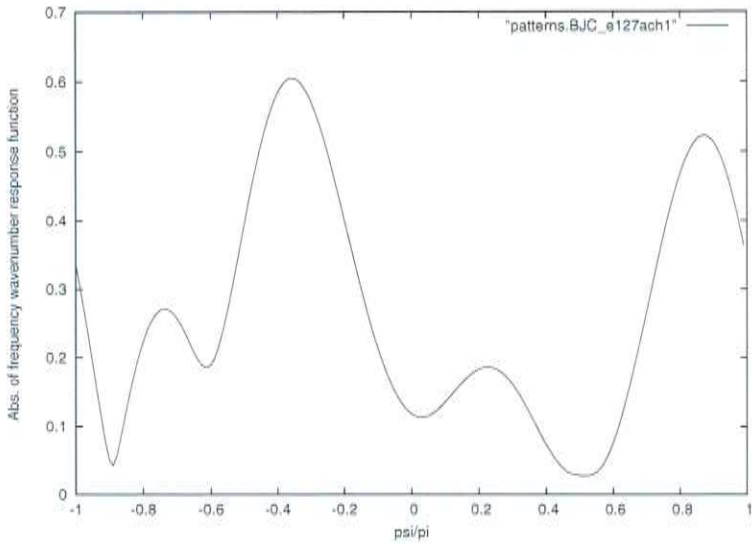
We added interference of -5 dB to the far field recording and re-run the beamforming experiments, using the best settings from our earlier evaluations. Using the old decoder, the WER for the DS beamformer increased from 58.35% to 63.74%. WER for the MVDR beamformer increased from 57.83% to 64.17%, thus the performance is below the baseline of the DS beamformer. For the HMM global optimization beamformer with adaptive cepstral means, the WER increased from 58.43% to 62.62% which is 1.12% absolute better than the DS beamformer and 1.55% better than the MVDR beamformer. When switching to the new decoder, the DS beamformer produced a WER or 58.36%, performing worse than the MVDR beamformer, which generated a WER of 58.26%. Again, the HMM beamformer performed best with a WER of 56.89%. Table 5.6 summarizes the experiments.

5.3 Summary

Without additional interferences, the MVDR beamformer first seemed to produce slightly better results than the HMM global optimization beamformer (about 0.1% absolute gain). Both adaptive beamformers performed better than the non-adaptive DS beamformer as long as the HMM beamformer could use the global optimization scheme. Using the iterative HMM beamformer without diagonal loading resulted in a performance, comparable to the one of the DS beamformer. It is very likely though, that its performance will increase to the level of the global optimization algorithm as soon as diagonal loading is correctly applied.

Further experiments demonstrated though, that the impact of the ASR system is large compared to the relatively small differences between the three

Figure 5.2: Beampattern of the HMM beamformer



beamformers, especially under good acoustic conditions. We could show, that only a few changes in the decoder result in big differences in the performance. Using the new decoder, we observed, that the differences between the beamforming algorithms shrunk to a minimum of 0.2% absolute difference in WER between the best and the worst algorithm.

The situation changed when we added an interferer to the speech signal. While the WER for all algorithms increased significantly, we could observe a superior performance of the HMM beamformer on both decoding systems. Compared to the next best system, we saw a gain in performance of approximately 1%. If we examine the beampattern of the HMM beamformer in Figure 5.2, we can see, that it tries to steer the main lobe towards the speaker at approximately $\psi/\pi = -0.4$ and to suppress side lobes in the region of the interferer at $\psi/\pi = 0.3$. Thus, even without information about the position of the interference, the beamformer could adapt to it, which lead to a superior performance compared to a DS beamformer with a static beampattern.

Overall, both adaptive beamformers could produce better results than the non-adaptive DS beamformer, but without additive interference, the differences are very small.

Chapter 6

Summary and Conclusion

6.1 Summary

The goal of this diploma thesis was the evaluation of adaptive beamformers for ASR systems. We presented three algorithms:

1. The non-adaptive DS beamformer, using just the speakers position to steer the main lobe of a beampattern.

2. The adaptive MVDR beamformer, which tries to adapt to noise and interferences during silence regions in the recording.

3. The adaptive HMM beamformer, a new approach we implemented for the first time, that uses the feedback of an ASR system to optimize the beamforming weights.

In order to derive an adaptive algorithm for the MVDR- and HMM beamformer, we introduced the basic concepts of Kalman filters and presented the square root information filter as a numerically stable solution to update the beam weights. Using the information filter, we could also directly implement diagonal loading; a method to prevent an unlimited growing of the beamforming weights due to errors in the calculation of the speakers position and array perturbations.

In order to evaluate the different algorithms, we distinguished two scenarios: In the first scenario, we used the ESST test set without additional interference to examine the performance of the beamforming algorithms under fairly good acoustic conditions. We could show that the performance of

the adaptive beamformers is slightly better than the one of the DS beamformer. While the MVDR beamformer seemed to perform best at a first glance, we could observe, that this lead was reduced significantly when we switched to another decoder. The performance of the HMM beamformer was highly depending on the amount of information available during the training. While the global optimization algorithm produced results comparable to the MVDR beamformer, results of the iterative HMM beamforming approach were just on the level of the DS beamformer.

In the second scenario, an additional interferer was added to the speech recordings to simulate cross-talk in a far field speech recording. As the adaptive beamformers are specially designed to deal with such interferences, we expected to see a bigger difference between the non-adaptive DS beamformer and both adaptive algorithms. Surprisingly, only the HMM beamformer could benefit from its adaptive design, while the MVDR beamformer performed even worse than the DS algorithm. We believe, that the MVDR beamformer fails to outperform the DS beamformer, because it assumes that signal and noise are uncorrelated. Given a reverberation time of approximately 350 ms in our experimental room setup, this assumption is obviously not true. The HMM beamformer on the other hand doesn't make any assumptions about the signal but uses only the ASR system to adapt its beam weights. By examine the HMM beampattern, we could perfectly see, how the beamformer tried to focus on the speaker while suppressing the signal from the direction of the interference.

6.2 Conclusion

Beamforming in general can significantly improve the performance of ASR systems. Without interferences, a slightly better performance of the adaptive beamformers compared to the non-adaptive DS beamformers is observable, but noticing the heavy influence of the ASR system to the WER, these differences might be below statistical significance. Evaluating the beamformers with an additive interferer, we could show, that only the HMM beamformer was able to reduce the influence of the interferer significantly, while the adaptive MVDR couldn't outperform the DS beamformer.

Beside the performance of the different beamformers on ASR systems, the choice for one or another beamformer is also depending on its computational complexity and the amount of information, that is available to train the

beamformer:

- The DS beamformer is the fastest algorithm that needs the least amount of information and no training of the beamforming weights. Given a position estimation of the speaker, this algorithm is capable of doing realtime beamforming even with a large number of microphones. This makes this algorithm an excellent choice, when a lot of microphones or only a source localizer are available and realtime computing is necessary.

- The MVDR beamformer needs the position estimation of a speaker and a speech activity detector for an accurate weight estimation. Especially the speech activity detection might become problematic for this algorithm, as it is very likely to fail in far field recordings. Even with only a few errors in the speech activity detection, the performance of the MVDR beamformer will decrease. Taking into account, that the difference in WER between DS- and MVDR beamformer, even under perfect conditions, is low, we can hardly imagine a situation, where this beamformer should be preferred.

- The HMM beamformer has the highest computational complexity. Even with an average number of microphones, this algorithm will hardly work in realtime on the currently available PCs. It needs a position estimation of the speaker and an ASR system which already performs reasonably well on the given speech data, in order to produce accurate cepstral means and variances. Without oracle state alignments, the performance of the algorithm is likely to decrease similar to the observations made in [Seltzer03], hence, given a high number of microphones and/or good acoustic conditions, the DS beamformer should be preferred.
  Advantages arise, when the speech recordings are completely available to perform a global optimization. Especially if the recordings are heavily disturbed and the amount of time to perform the training is less important, this algorithm may be the best choice.

6.3 Future Work

As the impact of the ASR system to the final result is undeniably high, further investigations are necessary to find standards for beamforming ex-

periments in order to get statistically significant results, when comparing different algorithms between different research groups.

The good performance of the HMM beamformer on the distorted signal indicates its use for highly disturbed far-field signals. To further examine the good performance of the HMM global optimization beamformer, additional experiments with moving speakers, cross-talk and non-oracle state alignments could be of interest. The CHIL database[1], consisting of different seminars, talks and meetings under realistic conditions, might for example be a good choice for further studies.

[1]CHIL, Computers in the Human Interaction Loop

Appendix A

Filterbanks

This appendix introduces the concept of filterbanks and their usage in beamforming algorithms. In our beamforming application, we use two filterbank modules to process a signal: An *analysis filterbank* and a *synthesis filterbank*. The analysis filterbank divides an incoming signal into several subbands, the synthesis filterbank on the other hand takes the beam-formed subbands and restores the original signal. A cascade of an analysis filterbank, a beamformer, and a synthesis filterbank is shown in Figure A.1.

Figure A.1: Analysis and synthesis filterbanks

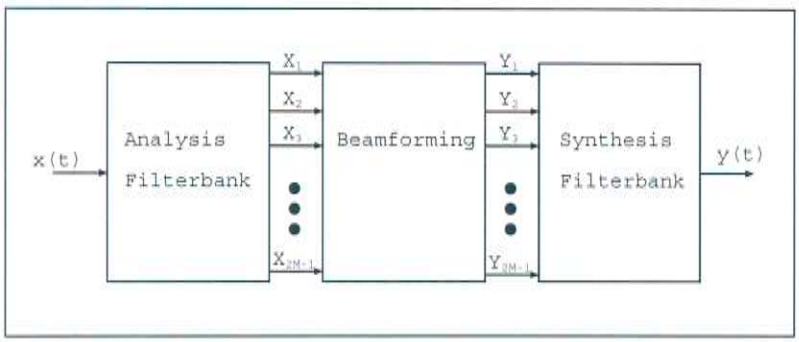
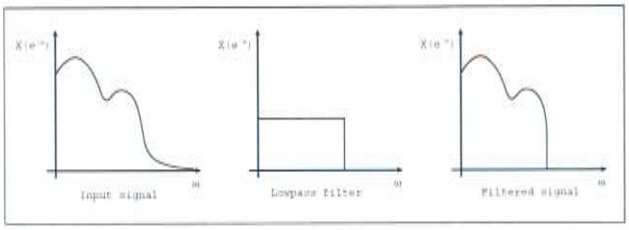


Figure A.2: Signal and filtered signal



A.1 Properties of Analysis and Synthesis Filterbanks

The analysis filterbank processes an incoming broadband signal and divides it into narrowband subbands. Dividing a signal into several subbands has three advantages for beamforming:

1. We can down-sample the subbands, which reduces the complexity of many necessary computations.

2. The subbands are narrowband, thus we can use beamforming techniques designed for narrowband signals instead of dealing with a broadband signal.

3. We can optimize the weights for all subbands independently.

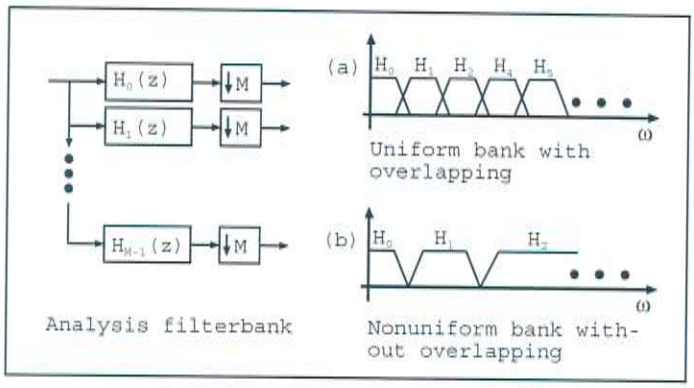
Let us explain these three points and their influence on our beamforming algorithms.

The necessary sampling rate to avoid aliasing depends on the highest signal frequency and is given by the Nyquist theorem, that states:

$$s \geq 2\omega_{max},$$

where s is the sampling rate and ω_{max} is the highest frequency of the recorded signal. In standard speech recordings, a sampling rate limited to 16 or 32 kHz is usually sufficient as most relevant parts of speech are concentrated in the lower frequency regions. To avoid aliasing we therefore have to *band-limit* our incoming signal with a *lowpass* filter to a maximum frequency of e.g. 8 or 16 kHz as shown in Figure A.2.

Figure A.3: M-channel analysis filterbank [Vaidya93]



Instead of using one lowpass filter to band-limit the signal, we can also use multiple bandpass *analysis filter* to divide the signal into several subbands with smaller bandwidth. A cascade of these filters and a unit to down-sample the subbands is what we call an analysis filterbank. Figure A.3 shows different possibilities how the analysis filters can be designed.

Instead of having one signal with e.g. $\omega_{max} = 8$ kHz, we now have for example four subbands with a bandwidth of $\omega_{max} = 2$ kHz each, thus we can reduce the sample rate for each subband to $s \geq 4$ kHz instead of having to use $s \geq 16$ kHz. A set of sub samples is normally called a *snapshot*.

Using subband samples with reduces sample rate usually leads to reduced computational complexity for vector and matrix operations. For example when doing an FFT, the FFT filter length depends on the sampling rate and the length of a windowing function. With a hamming window length of 32 ms and a sampling rate of 16 kHz we need a filter length of at least

$$\frac{16000}{32} = 500.$$

For an FFT the filter length has to be 2^M, hence we choose a filter length of 512. The complexity of an FFT is $O\left(N \ \log_2(N)\right)$, thus with $N = 512$ we have approximately

$$512 \ \log_2(512) = 4608$$

operations. With a sampling rate of 4 kHz for our four subbands we need an FFT filter length of 128 ($\frac{4000}{32} = 125$). This reduces the necessary operations for one FFT to

$$128 \ \log_2(128) = 896.$$

On the other hand, we now have to compute the FFT four times (one time for each subband). Still the total reduction of operations becomes

$$4608 - 4 \cdot 896 = 4608 - 3584 = 1024,$$

which is a reduction of more than 25%.

Despite the computational savings we now have narrowband subbands instead of one broadband signal. However if we want to use beamforming algorithms specially designed for the narrowband case, we require that the subbands can be processed independently. As proved in [VanTrees02, Chapter 5], the subbands $X(\omega_m, k)$ are joint circular complex Gaussian random vectors if we assume the input signal $x(t)$ to be a real vector Gaussian random process. This means, that the joint densities of $X(\omega_m, k)$ for different m and k is equal to the product of the individual densities. Thus, we can process the subbands independently and therefore make use of narrowband beamforming algorithms.

The synthesis filterbank is the counterpart to the analysis filterbank; it up-samples the different subbands and combines them to rebuild a single broadband signal. A careful choice of synthesis filters is necessary to achieve the *perfect reconstruction* (PR) property and is depending on the analysis filters as will be shown in the next section.

A.2 Perfect Reconstruction Filterbanks

The PR property states that for a cascade of an analysis and synthesis filterbank, the reconstructed output signal of the synthesis filterbank is equal to the input signal of the analysis filterbank. As stated in [Vaidya93, Chap. 5], three problems may arise, if we do not carefully design the filterbanks, namely *aliasing*, *amplitude distortion* and *phase distortion*.

Amplitude distortion arises, when the filter is not all-pass for all frequencies. We can compensate amplitude distortion by designing filters pairwise such that the effect of amplitude distortion is canceled out. Phase distortion arises, when the filters do not have linear phase. We can avoid this problem by using *finite impulse response* (FIR) filter, where a linear phase can be guaranteed. Aliasing arises, as analysis filters do not have zero transition bandwidth and stop band gain in practical applications. The signal $x(t)$ is therefore not band-limited and down sampling inside the analysis filterbank results in aliasing.

The design of a filter with sharp transitions is very expensive, hence we may have to use analysis filters that are overlapping (Figure A.3(a)). That means that each subband can have substantial energy for a bandwidth exceeding the ideal pass band region. If we design non-overlapping filter as in Figure A.3(b), we get an attenuation of the input signal between two filters. Boosting this frequency region can compensate this effect but will also result in an amplification of noise. Fortunately aliasing can be avoided by designing the synthesis filterbank with respect to the analysis filterbank.

Cosine Modulated Filterbanks

The *cosine modulated filterbank* (CMF) is one example for a PR filterbank, where the analysis and synthesis filterbank are designed to avoid aliasing, amplitude-, and phase distortion. It was independently developed by Malvar [Malvar90, Malvar91], Ramstad [Ramstad91] and Koilpillai and Vaidyanathan [Koilpillai91, Koilpillai92]. The CMF only requires the design of one *prototype filter*, which is used to form all analysis and synthesis filters. As stated in [Vaidya93, Chap. 8.5.5], the CMF has two outstanding advantages:

1. "The cost of the analysis bank is equal to that of one filter, plus modulation overhead. The modulation itself can be done by fast techniques such as the fast discrete cosine transform (DCT). See, for example, [Yip87]. The synthesis filters have the same cost as the analysis filters."

2. "During the design phase, where we optimize the filter coefficients, the number of parameters to be optimized is very small because only the prototype has to be optimized."

The filterbanks are designed, in the effective *polyphase* structure. The input of the analysis filterbank is a stacked vector $x(t) \epsilon \mathbb{R}^{2M \times 1}$,

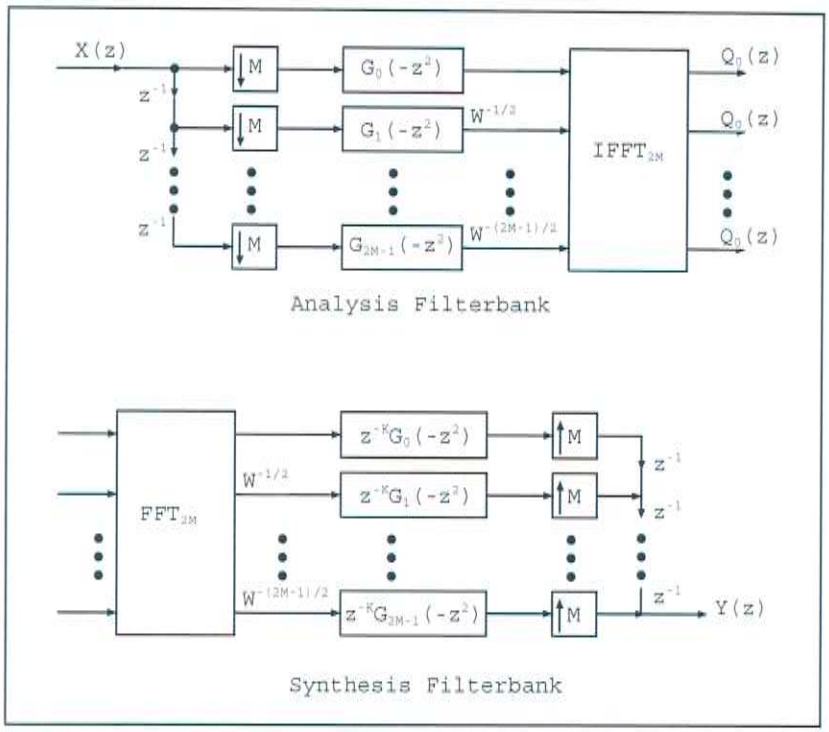
$$x(0) = \begin{pmatrix} x_{M-1}(0) \\ \vdots \\ x_0(0) \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \; x(1) = \begin{pmatrix} x_{M-1}(1) \\ \vdots \\ x_0(1) \\ x_{M-1}(0) \\ \vdots \\ x_0(0) \end{pmatrix}, \; x(2) = \begin{pmatrix} x_{3M-1}(2) \\ \vdots \\ x_{2M}(2) \\ x_{2M-1}(1) \\ \vdots \\ x_M(1) \end{pmatrix}, \ldots$$

the output is similarly given by a stacked output vector $y(t)\epsilon\mathbb{R}^{2L\times 1}$, with

$$y(0) = \begin{pmatrix} y_{L-1}(0) \\ \vdots \\ y_0(0) \\ 0 \\ \vdots \\ 0 \end{pmatrix}, y(1) = \begin{pmatrix} y_{L-1}(1) \\ \vdots \\ y_0(1) \\ y_{L-1}(0) \\ \vdots \\ y_0(0) \end{pmatrix}, y(2) = \begin{pmatrix} y_{L-1}(2) \\ \vdots \\ y_0(2) \\ y_{L-1}(1) \\ \vdots \\ y_0(1) \end{pmatrix}, \ldots$$

Given G_a, the polyphase matrices of the analysis filters, the complete analysis filterbank is given by a delay chain followed by down sampling units, the analysis filters G_a and an inverse FFT. The synthesis filterbank is equivalently designed: After performing an FFT, the signals are processed by the synthesis filter in polyphase structure G_s and up-sampled. The final output is obtained by reconstructing the signal over a delay chain. Figure A.4 shows the analysis and synthesis filterbank in polyphase structure.

Figure A.4: Cosine modulated analysis and synthesis filterbank [McDonough05]



Bibliography

[Davis80] S. Davis, P. Mermelstein, "Comparison of Parametric Representation for Monosyllablic Word Recognition in Continously Spoken Sentences", IEEE Trans. on Acoustic Speech and Signal Proc., Vol 28, pp. 357-366, 1980

[Dyer69] P. Dyer and S. McReynolds, "Extension of Square-Root Filtering to Include Process Noise", Journal of Optimization Theory and Applications, Vol. 3, No. 6, 1969

[Fraser67] D. C. Fraser, "A New Technique for the Optimal Smoothing of data", M.I.T. Instrumentation Lab., Rep. T-474, Jan. 1967

[Griffiths82] L. J. Griffiths & C. W. Jim, "An Alternative Approach to Linearly Constrained Adaptive Beamforming", IEEE Trans. on Antennas and Propagation, AP-30, pp. 27-34, 1982M. Hou, R. J. Patton, "Optimal Filtering for Systems with Unknown Inputs", IEEE Trans. on Automatic Control, Vol 43, pp. 445-449, 1998

[Haykin96] S. Haykin, "Adaptive Filter Theory", Prentice-Hall, NJ, 1996

[Johnson93] D. H. Johnson, D. E. Dudgeon, "Array Signal Processing", Prentice Hall, New Jersey, 1993

[Kailath68] T. Kailath, "An Innovations Approach to Least-Squares Estimation - Part I: Linear Filtering in Additive White Noise", IEEE Trans. on Automatic Control, Vol. 13, pp. 646-655, 1968

[Kaminski71] P. Kaminski, A. Bryson, S. Schmidt, "Discrete Square Root Filtering: A Survey of Current Techniques", IEEE Trans. on Automatic Control, Vol 16, pp. 727-736, 1971

[Malvar90] H. S. Malvar, "Modulated QMF Filter Banks with Perfect Reconstruction", Electronics letters, vol. 26, pp. 906-907, 1990

[Malvar91] H. S. Malvar, "Extended Lapped Transforms: Fast Algorithms and Applications", Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Proc., pp. 1797-1800, Toronto, Canada, 1991

[McDonough04] J. McDonough, D. Raub, M. Wölfel, A. Waibel, "Towards Adaptive Hidden Markov Model Beamformers, Proc. NIST ICASSP Meeting Recognition Workshop, Canada, 2004

[McDonough05] J. McDonough, U. Klee, A. Schkarbanenko, "Maximum Likelihood Beamforming with Cosine Modulated Filter Banks", Technical Report 106, Karlsruhe, Germany, 2005

[Rabiner86] L. R. Rabiner, B. H. Juang, "An Introduction to Hidden Markov Models", IEEE ASSP Magazine, January 1986

[Rabiner93] L. R. Rabiner, B. H. Juang, "Fundamentals of Speech Recognition", Prentice Hall, New Jersey

[Ramstad91] T. A. Ramstad, "Cosine Modulated Analysis-Synthesis Filter Bank with Critical Sampling and Perfect Reconstruction", Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Proc., pp. 1789-1792, Toronto, Canada, 1991

[Steinhardt88] A. O. Steinhardt, "Householder Transformation in Signal Processing", IEEE ASSP Magazine, pp. 4-12, July 1988

[Koilpillai91] R. D. Koilpillai, P. P. Vaidyanathan, "New Results on Cosine-Modulated FIR Filter Banks Satisfying Perfect Reconstruction", Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Proc., pp. 1793-1796, Toronto, Canada, 1991

[Koilpillai92] R. D. Koilpillai, P. P. Vaidyanathan, "Cosine-Modulated FIR Filter Banks Satisfying Perfect Reconstruction", IEEE Trans. on Signal Processing, vol. SP-40, pp. 770-783, 1992

[Raub04] D. Raub, J. McDonough, M. Wölfel, "A Cepstral Domain Maximum Likelihood Beamformer for Speech Recognition", Proceedings, Interspeech, Korea, 2004

[Seltzer03] M. L. Seltzer, "Microphone Array Processing for Robust Speech Recognition", Ph.D. dissertation, Carnegie Mellon University, PA, 2003

[Vaidya93] P. P. Vaidyanathan, "Multirate Systems and Filter Banks", Prentice-Hall, Englewood Cliffs, NJ, 1993

[VanTrees02] H. L. Van Trees, "Optimum Array Processing", Wiley & Sons, NY, 2002

[Viterbi67] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", IEEE Trans. on Information Theory, Vol. 13, pp. 260-269, 1967

[Yip87] P. Yip and K. R. Rao, "Fast Discrete Transforms", Handbook of digital signal processing, Academic Press, San Diego, CA, 1987